TAPR

38th

ARRL and TAPR
Digital Communications Conference

Detroit, Michigan
September 20-22, 2019

38th ARRL and TAPR Digital Communications Conference

ARRL
225 Main Street
Newington, CT 06111-1494 USA
tel: 860-594-0200   www.arrl.org

Tucson Amateur Packet Radio
PO Box 852754
Richardson, TX 75085-2754 USA
tel: 972-671-8277   www.tapr.org

ARRL and TAPR – 38th Annual
Digital Communication Conference
September 20-22, 2019 – Detroit, MI

Greetings!

Welcome to the 38th annual ARRL and TAPR Digital Communications Conference (DCC).

Each year a call for papers is announced. In August we collect the papers and print the proceedings that you are holding in your hands. This is a great testament to the mission of "advancing the amateur radio art." Ideas are shared which in turn create more new ideas. Each DCC builds upon the previous. For a complete listing of past proceedings, please see **https://www.tapr.org/pub_dcc.html**

Since 2008 we've had the pleasure and video skills of Gary Pearce, KN4AQ, of HamRadioNow. He and a small team of volunteers video recorded and edited each of the DCCs including our banquet speaker and Sunday seminar. If you would like to virtually attend DCC please see **https://www.hamradionow.tv/tapr-dcc.**

As of 2018, Gary turned over the video reins to Jason Johnston, KC5HWB, who produces "Ham Radio 2.0." Jason returns again this year and we are very grateful to him in broadcasting these exquisite talks to the world. You can view Jason's videos at **https://www.livefromthehamshack.tv/?s=dcc**. Please support Jason on Patron (**https://www.patreon.com/HamRadio**) and Kickstarter (**https://www.kickstarter.com/profile/hamradio20**) campaigns to enable him to travel to and record each DCC.

All of the video recording efforts and printed proceedings records for posterity all the fine work these experimenters have done. While attending virtually is educational, we are certain that you will enjoy the sharing of ideas and socializing that in-person attendance of the DCC can give. Please tell others about the experience you had here and help grow attendance for the DCC.

With all of that said, please sit back, relax, and enjoy this years DCC.

73,

Steven Bible, N7HPR
President, TAPR

Schedule at a Glance

Thursday, 19 Sep

9:00 AM	TAPR Board Meeting
5:00 PM	(everyone is welcome to attend)

Friday, 20 Sep

8:00 AM	Conference Registration and Demonstration Room Open
8:45 AM	Welcome
9:00 AM	Technical Presentations
Noon	Lunch
1:00 PM	Technical Presentations
5:30 PM	Friday Night Social
10:00 PM	Demonstration Room Closed

Saturday, 21 Sep

8:00 AM	Conference Registration and Demonstration Room Open
8:45 AM	Welcome
9:00 AM	Technical Presentations
Noon	Lunch
1:00 PM	Technical Presentations
4:45 PM	TAPR Membership Meeting
6:00 PM	No Host Cash Bar
7:00 PM	Dinner Banquet
10:00 PM	Demonstration Room Closed

Sunday, 22 Sep

8:00 AM	Sunday Seminar
Noon	

Rooms at a Glance

Registration – **tbd**
Demonstration Room – **Dearborn**

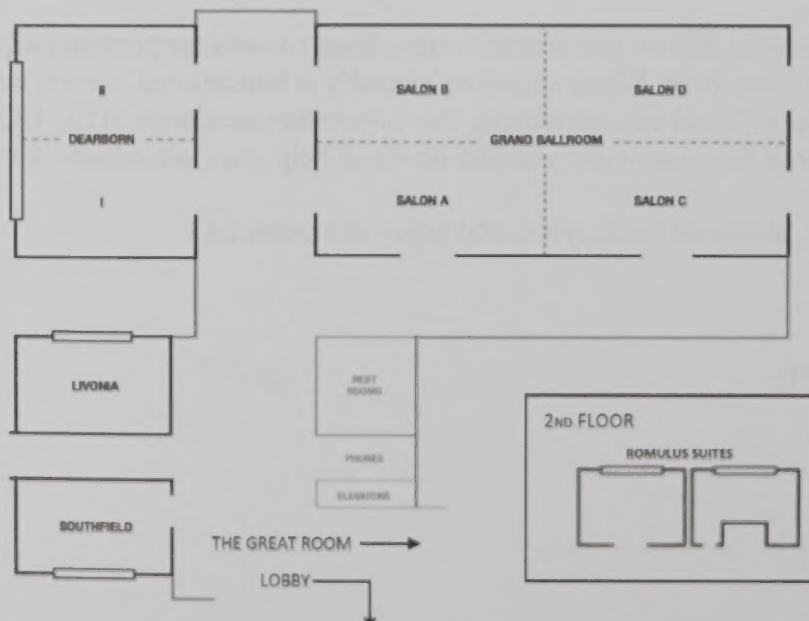**Thursday**
TAPR Board Meeting – **Romulus**

Friday
Main Session Technical Presentations – **Ballroom**
Lunch – **North Private**
DCC Social – **North Private**

Saturday
Main Session Technical Presentations – **Ballroom**
Introductory Sessions Presentations – **Romulus (2nd Floor)**
Lunch – **North Private**
Dinner Banquet – **North Private**

Sunday
Sunday Seminar – **Ballroom**

Preliminary 2019 DCC Conference Schedule

	Friday	Saturday		Sunday
8:00 AM	**Conference Registration** Demonstration Room Open	**Conference Registration** Demonstration Room Open		Sunday Seminar **8:00 AM – 12:00 AM**
8:45 AM	Main Session **Welcome and Introductions**	Main Session **Welcome and Introductions**	Introductory Session	**Learn to build and operate your own SatNOGS ground station**
9:00 AM	F1	S1	N1	Dan White, AD0CQ Corey Shields, KB9JHU
9:45 AM	F2	S2		
10:30 AM	F3	S3	N2	
11:15 AM	F4	S4		
Noon	Lunch	Lunch		
1:00 PM	**Lightning Talks** (Impromptu 5-minute talks)	**Lightning Talks** (Impromptu 5-minute talks)	N3	
1:45 PM	F5	S5		
2:30 PM	F6	S6	N4	
3:15 PM	F7	S7		
4:00 PM	F8	S8		
4:45 PM	**Play Time** in the Demonstration Room	**TAPR Annual Meeting**		
5:30 PM	Friday Night Social **No-Host Cash Bar**	**Play Time** in the Demonstration Room		
6:00 PM		Dinner **No-Host Cash Bar (6:00 PM)** **Dinner (7:00 PM)** Bill Brown, WB8ELK **Awards Presentation** **Prize Drawings**		
10:00 PM	**Demonstration Room Closed**			

Table of Contents

PSAT2 DTMF Experiment
APRStt – Touchtone® Digital Communications
Using any Radio for Data Exchange

Bob Bruninga, WB4APR
US Naval Academy Satellite Lab

DTMF (Dual Tone Multi Frequency) (aka Touchtone®) signaling has been built into almost all Amateur Radios for decades and more recently cellphones, yet most hams do not make use of this powerful data entry and data exchange capability for their special events. They continue to use laborious voice directed nets and hand transcribed data. Although APRS brings an ideal digital communications capability to such applications, not all volunteers have the more expensive APRS capable radios. But they all have DTMF!
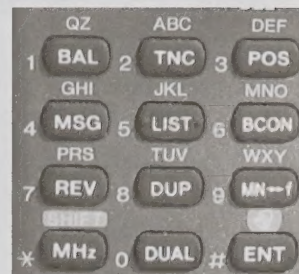
APRStt has been around since 2001 [*http://aprs.org/aprstt.html*] and used by the author for several special events, but it has not caught on, even though now HT's with built in DTMF are down around $50 each, a fraction of the cost of an APRS radio. To wake things up and give a worldwide demonstration of the power and practicality of DTMF signaling for data exchange, the new PSAT2, on orbit since June 2019, has a primary mode for DTMF uplink and voice/APRS downlink. Although an entire APRStt protocol has been developed for almost any application that APRS can do, the implementation on PSAT2 consists of two DTMF formats that are practical in the space environment. One for worldwide position reporting by Grid Square and the other for sending APRS messages. [*http://aprs.org/psat2.html*]
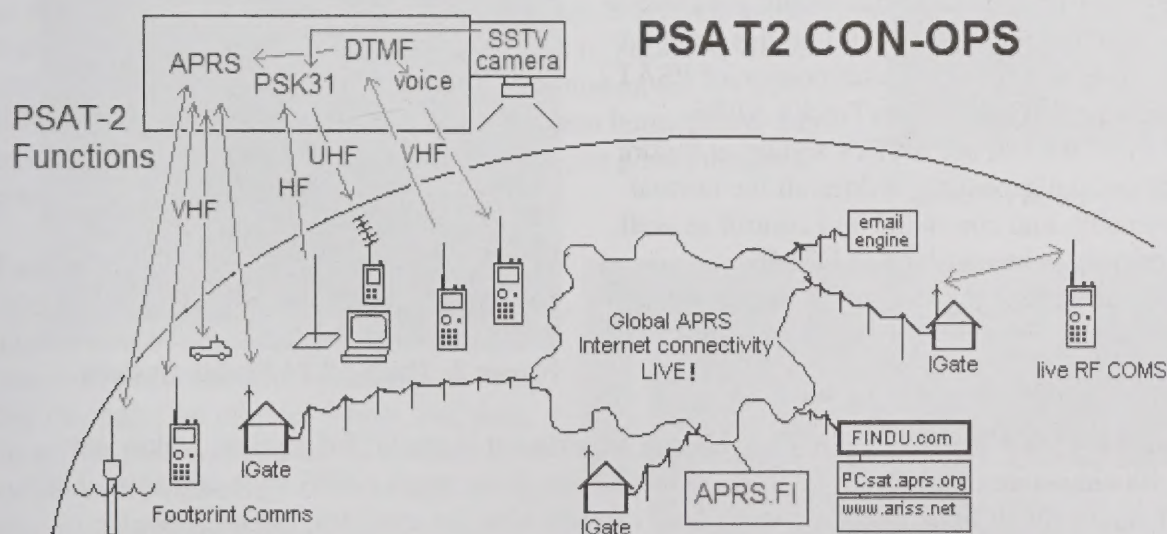

Figure 1. PSAT2 has Packet, PSK31, SSTV, Voice Synthesis and DTMF transponders.

As you can see in the Concept of Operations Figure 1, PSAT2 carries a number of experiments for a number of additional modes beyond just simple digipeating. In fact, these other unique

modes have higher priority compared to digipeating because they are new and different. These modes are a full duplex PSK31 transponder with HF SSB uplink and UHF FM audio waterfall downlink. Also included in the downlink audio passband every four minutes is an SSTV image either from memory or from the live camera if the view appears to be the Earth (ie, not full sun and not the blackness of space). And importantly for this paper, the first ever DTMF uplink-to-APRS downlink with Voice user feedback.

PSK31 and SSTV: The PSAT2 PSK31 and SSTV downlink is on 435.350 +/- Doppler using any UHF FM receiver (SSTV only when PSAT2 is in the Sun). You will see the audio waterfall with satellite telemetry at around 280 Hz, PSK31 users between 550 to 950 Hz and occasional SSTV images between 1200 to 2300 Hz. Using SSB uplink on ten meters and UHF FM downlink, the total Doppler shift of PSK31 users is only about 1 Hz per second average. On approaching and receding passes for a user, where Doppler changes very little, conventional PSK31 decoders can work OK.

Full Duplex, Doppler Corrected Uplink: But for serious two-way contacts, users should download the special Doppler corrected PSK31 uplink program written by Andy Flowers, K0XY. It not only adjusts your uplink Doppler automatically so you stay at a fixed location in the passband, but it also allows you to operate full duplex with your HF uplink for the full duration of the pass. You can chat with everyone in parallel as fast as you can type! Download: http://www.frontiernet.net/~aflowers/dopplerpsk/dopplerpsk.html

Packet Digipeater: PSAT2 becomes the 6th active APRS digipeater in space. The goal is to have enough digipeaters so that Ham users around the world are never more than a few minutes from position/status reporting and 2-way text messaging anywhere on Earth. Ten satellites would get close to users being no more than about 10 minutes from any next pass. The basis of the packet portion of PSAT2 is the re-packaged Byonics TinyTrack4 APRS transceiver that we call the SATT4 shown in Figure 2. In addition to digipeating, it does all the normal packet telemetry, and command and control as well. Initially, priority is being given to DTMF experiments and the digipeater is off. For live status and downlinks, see [*http://aprs.org/sats.html*].
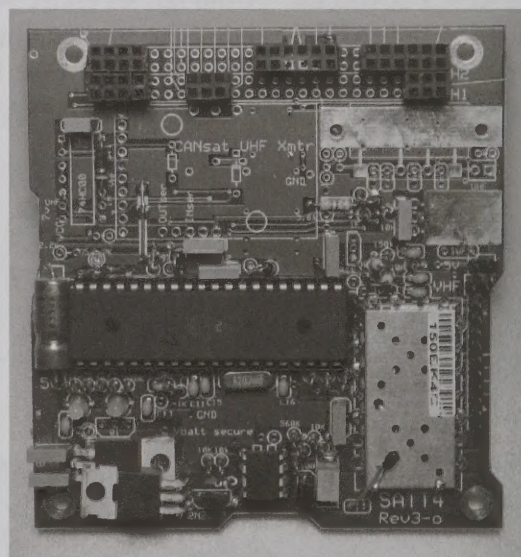
Figure 2. The SATT4 Packet System

APRStt and DTMF: But the real fun and unique experiment is the DTMF uplink, which lets anyone with almost any inexpensive FM radio participate in the usual APRS style contacts. This mission is not so much to demonstrate some kind of great idea for satellites, but more so, to demonstrate to APRS users worldwide the power of DTMF data from-any-radio into the global APRS-IS (internet system). The DTMF downlink from PSAT2 is visible on this web page: http://www.aprsat.com/dtmf

Attitude Sensors and Control
board with Basic Stamp
and uMag3

Brno U PSK31 with
29 MHz uplink
435 MHz down

Brno U SSTV camera

1 2 kg Ballast for
long orbit life

NiCd Battery Board

Touchtone uplink and
Voice Downlink

Byonics/USNA SATT4
VHF APRS
Transponder

Nitinol Whip Antenna

90 degree VHF
phasing harness

PSAT2 - US Naval Academy Student Project (and Brno University)

Figure 3. The PSAT2 internal and external views. The camera peeps through a hole in the rail.

Operating Pecularities: PSAT2's 28 degree inclined elliptical orbit with apogee at 860 km and low perigee at 300 km means it never gets higher than 28 degree latitude which ma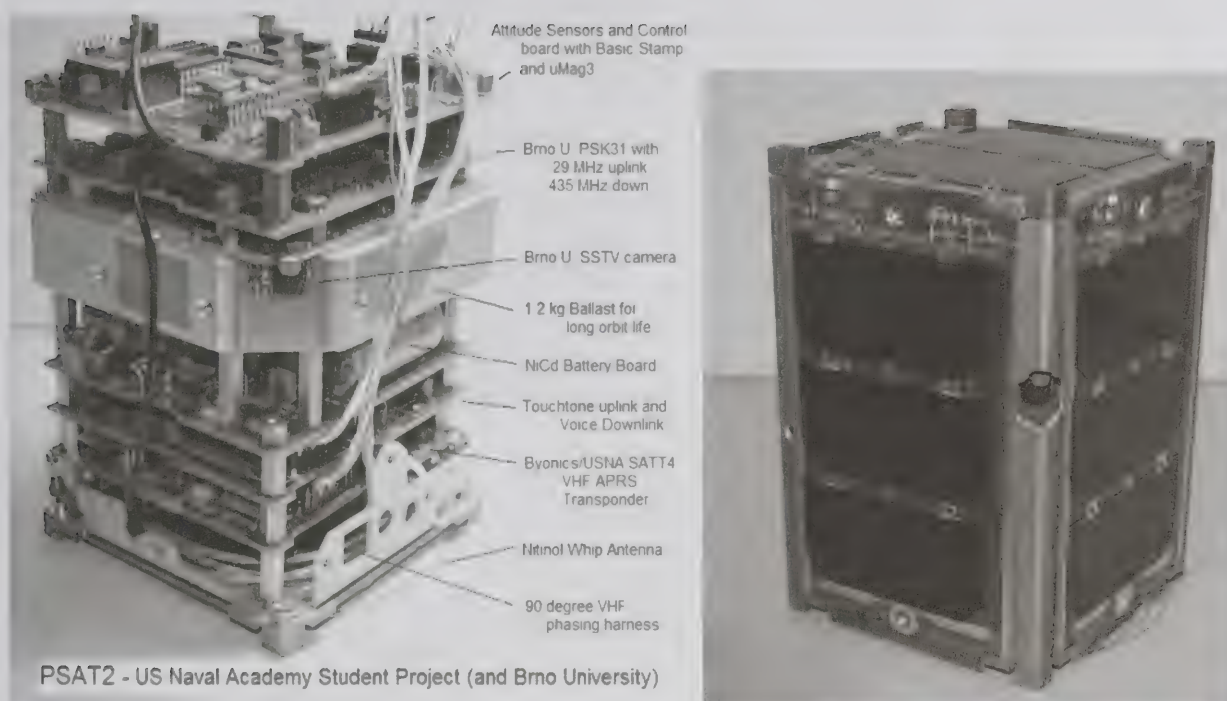kes it difficult to work in Northern states. But the significant difference is the footprint between apogee and perigee shown in Figure 4, which can make a 20 degree or so elevation difference on the horizon. When apogee circulates to be over the northern hemisphere, then more northern stations can work it. The apogee goes through a complete cycle every 34 days.
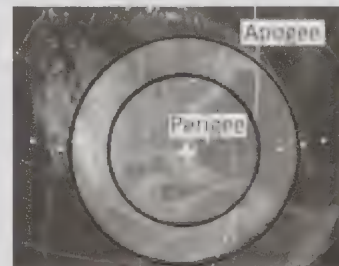
Figure 4. Footprint sizes

Pass Timing for Mobiles: Another interesting thing about the orbit is that it is almost time synchronous; meaning that a pass will occur almost the same time every day (though five minutes later). But then, an earlier pass will appear 90 minutes or so earlier every other day. This makes it very easy to do mobile/portable operations without any computer once you hear one pass and remember the offsets.
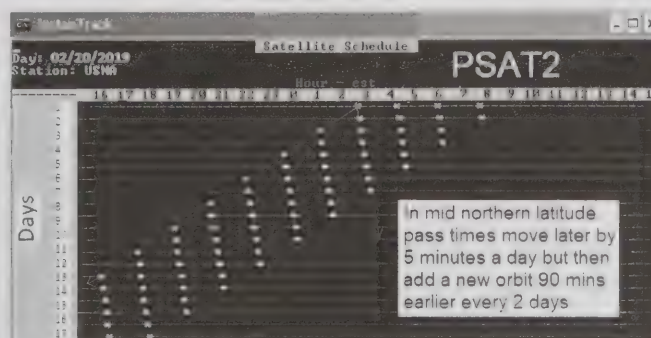
Figure 5. PSAT2's easy to remember pass times.

Camera: By default, the HF/UHF PSK31 and SSTV modes have been enabled from launch and available to users. During the initial activation, the camera took some images to file. It has been sending them down over time. The best earth photo is shown here. Later, live camera and other modes were enabled.

Figure 6. One of the best early SSTV camera images from the Brno University camera.

Mode Status Bits: The 8 bits on the end of every one minute telemetry packet indicates the status of the various modes. Normal digipeater operations will show a pattern of x1xx1xx. But if DTMF is enabled, the DTMF bits are 1101xxxx as shown here. Generally, PSAT2 will be in DTMF mode with the digipeater off to keep the DTMF uplink exclusive for DTMF users on 145.980 MHz.

Telemetry Control Bits

	DTMF ON	n/a	Reset CPUs	DTMF msgs	PSK31 suppress	DIGI ON	APRS-to-voice	SYSOP auth
T#SSS, . . . ,	0	1	0	0	0	0	0	0

Figure 7 Default mode Status Bits

Narrowband Uplink Modulation is Required: Unfortunately, PSAT2 uses a low cost off-the-shelf receiver with modern narrowband channelization. This causes distortion on all uplinks that are not Doppler compensated. Don't even think about transmitting to PSAT2 with normal USA model radio not set to narrowband and using 5KHz channel steps except for the center few seconds as the Doppler passes rapidly through zero offset. Do not transmit otherwise because PSAT2 will never decode you and you are only jamming others.

Operating Frequencies:

 - APRS/DTMF/Voice downlink: 145.825 1200 baud
 - APRS digi uplink 145.825 if enabled
 - DTMF uplink 145.980 when enabled
 - PSK31 Downlink: 435.350 MHz +/- 5 kHz FM (300 mw)
 - PSK31 Uplink: 29.4815 MHz PSK31 SSB (25 W and omni vertical typical)

Overall APRS Satellite live Web Page: http://aprs.org/sats.html

APRStt GRID MAP Encoding: The map and table below encodes the 99 grids that have 99% of the worlds ham population into only 4 DTMF digits by converting the first two letters into two digits from the 00-99 table. You can see that our Maryland FM19 grid becomes 1819 in DTMF. A table is also shown below that simplifies finding your grid. Notice that three grids for Indonisa OI, OJ, and PI appear in the Canada, USA and Central America blocks.

The table shows 00 thru 99 to give 4 digit Grids for the next APRStt DTMF satellite using DTMF only
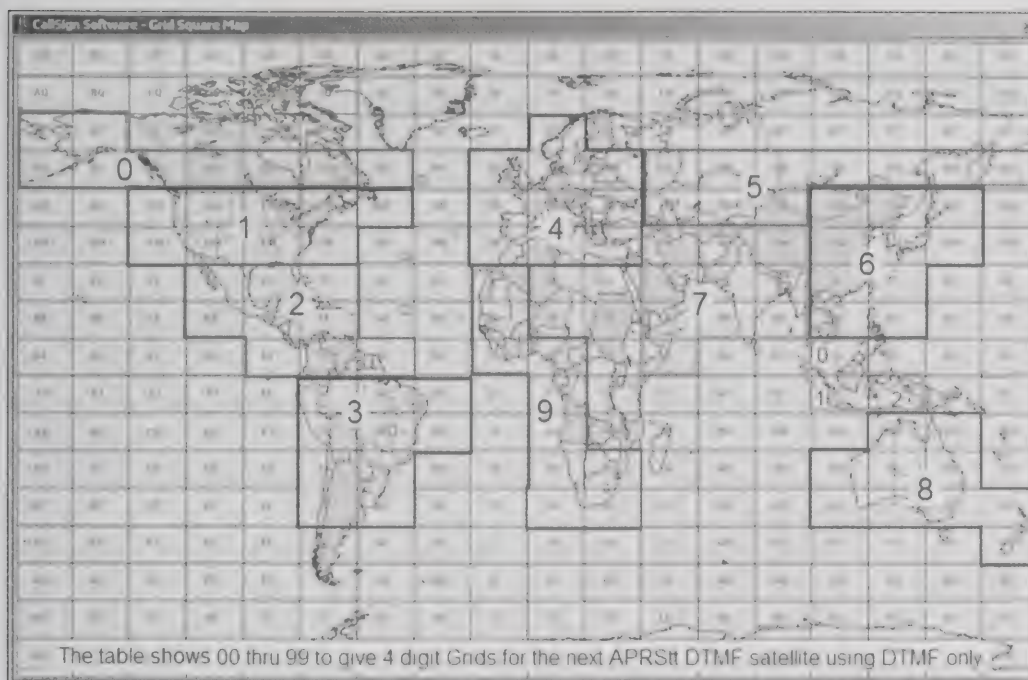
0x Canada
00 AP, BP, AO, BO, CO
05 DO, EO, FO, GO, OJ
1x USA
10 CN, DN, EN, FN, GN
15 CM, DM, EM, FM, OI
2x C America
20 DL, EL, FL, DK, EK
25 FK, EJ, FJ, GJ, PI
3x S. America
30 FI, GI, HI, FH, GH
35 HH, FG, GG, FF, GF
4x Europe
40 JP, IO, JO, KO, IN
45 JN, KN, IM, JM, KM
5x Russia
50 LO, MO, NO, OO, PO
55 QO, RO, LN, MN, NN
6x Japan, China
60 ON, PN, QN, OM, PM
65 QM, OL, PL, OK, PK
7x India
70 LM, MM, NM, LL, ML
75 NL, LK, MK, NK, LJ
8x Aus/NZ
80 PH, QH, OG, PG, QG
85 OF, PF, QF, RF, RE
9x Africa
90 IL, IK, IJ, JJ, JI,
95 JH, JG, KG, JF, KF

Other Grids: Due to popular demand for hams that were left off that map (ie, Hawaii), we can (with the efforts of WA7MXZ and KB6EBR) make adjustments. For Hawaii, we have designated the grid of HI and can distinguish it from the real HI in Brazil by comparing where PSAT2 is at the time of the contact and then correct for it on the DTMF downlink page.

Region		0	1	2	3	4	5	6	7	8	9
Canada Alaska	0	AP	BP	AO	BO	CO	DO	LO	FO	GO	OJ
United States	1	CN	DN	EN	FN	GN	CM	DM	FM	IM	OI
Central America	2	DL	EL	FL	DK	EK	FK	LJ	FJ	GJ	PI
South America	3	FI	GI	HI	FH	GH	HH	FG	GG	FF	GF
Europe	4	JP	IO	JO	KO	IN	JN	KN	IM	JM	KM
Russia	5	LO	MO	NO	OO	PO	QO	RO	LN	MN	NN
Japan /China	6	ON	PN	QN	OM	PM	QM	OL	PL	OK	PK
India, MidEast	7	LM	MM	NM	LL	ML	NL	LK	MK	NK	LJ
Australia/New	8	PH	QH	OG	PG	QG	OF	PF	QF	RF	RE
Africa	9	IL	IK	IJ	JJ	JI	JH	JG	KG	JI	KI

APRStt Callsign Encoding: PSAT2 cleverly compresses a 6 character call into only 10 digits following the 4 digit grid noted above. The first six digits of the call are the matching single keys for the callsign letters, EG: 924227 for WB4APR. The next 4 digits encode the 2 bit location of each of the 6 call letters on each of the 6 keys used. For example, the 6 letters of WB4APR on the 6 keys are key locations 120112. Since each location is between 0 and 3, they can be encoded in 2 bits each (powers of 4) and assembled left to right into a 12 bit binary number. To convert to decimal, take the first 2 bits times 1024, the next 2 bits times 256, the next 2 bits times 64, the next times 16, the next times 4 and the last 2 bits times 1. Then add them up and get the 4 digit decimal "key code" (1558 in this example).


```
FM 19 WB4APR
*18 19 9 24 277 1558 #
        1 20112
six 2 bit nos showing locations of 6 letters on keys
then converted to 4 digit decimal
                                    wb4apr
```

This 4 digit decimal number we call your callsign key code. For those that are 12-bit-challenged, Bob Wood WA7MXZ has written an *DTMF Callsign Encoder* to do this 4 digit number for you.

For shorter calls, right-pad to 6 with spaces. A space is encoded as the "0" key with the key location code of 1. The entire Grid and call report adds a "*" at the beginning and a "#" at the end for the full 16 DTMF key report.   See: [*http://aprs.org/PSAT2Translator.html*]

Other than your gridsquare, you only need to memorize the last four digit special code because your call is simply spelled out with the letters on the keys.

DTMF Robustness: The combined 16 key combination is then stored in the DTMF memory of the users radio so that it can be transmitted in a single 3 second burst. Since the entire code is self contained, is always 16 keys, is sent at a standard speed, always begins and ends with known keys and all keys in-between are decimal only, then any other combinations will be ignored. Also the usual failure mode of DTMF is duplication of digits or omission of digits which will be ignored by the mentioned constraints. A successful uplink will be ACK-ed by voice since the DTMF user cannot see the APRS downlink.

Hints: Since the hardest thing for a DTMF decoder to recognize is two of the same digits in a row, check your own code. If there are no duplicate digits in a row, then you *might* be able to select fast DTMF on your radio. But otherwise, we have found that slower DTMF works best (100 ms). The most important thing of course is using narrowband FM and tracking Doppler within a few hundred Hz. Also, the DTMF receiver goes to sleep after 5 minutes of no use. If you are the first user on your continent, press and hold a key for 3 seconds to wake it up.

DTMF MESSAGES: The DTMF decoder will also accept encoded Messages. Since everything in ham radio has already been said, we simply stored the top 99 common ham radio messages on the spacecraft and you select the appropriate message with a two-digit message number (00 to 99). These messages are the standard ARL radiograms plus some other special ones (such as 40,41 and 42 for QSL's) for this satellite. To send one of these messages in the same 16 key DTMF memory use either the C or B format. These are 16 key DTMF strings that begin with the *"C"* key to indicate a message, and then a 2 digit message number and then a 2 digit modifier xx, followed by the above encoded CALLSIGN. If the message is an actual ARL **Emergency message**, then the modifier should be 99 and the speech will include the word "EMERGENCY". If the modifer is anything over 90, then it will not say Emergency but will say TEST. If the modifier is less than 90, then the modifier will only be used if the template for that message has a blank in it for insertion of the modifier.

DTMF ARL Radiogram Message Format:	*CMMxxCCCCCXXXX#*
DTMF special reversed QSL Message Format:	*Bxx40CCCCCCXXXX#*

For messaging it is assumed that you will generally just key in the first 3 digits and then finish the uplink with the remaining stored callsign from memory. The **C** format is when you want to send multiple messages from your callsign. Type in the CMM and then have the rest stored in a DTMF memory. The **B** format is when you want to send the same message (such as one of the 3 QSL messages) but easily modify the modifier. In this case, you key in Bxx and then store the remainder of the message and call in a DTMF memory.

In either case, the *"xx"* digits are a numeric modifier that will replace any *"__"* blank in the message text. See the *Actual Flight list of messages* and for background the *standard ARL radiograms* and a copy of the *Maritime Emergency Codes* that are also included.

Making a Contact: When you send your grid and call by DTMF the spacecraft will say *"GRID FM19 from WB4APR, QSO number xx"*. The QSO number increments with each new station up to 99 and rolls over. To complete a contact with such a grid, send the QSL message number 40 (or 41,42) with the QSO number xx. The spacecraft will say *W3XYZ says message number 40 QSL your QSO number xx, my number is yy"* to complete the contact. Of course, an APRS copy of each of these messages will also come down on the downlink for those with APRS radios and be captured by the http://www.aprsat.com/dtmf page.

-------------------------------   **APRStt General Use**   ---

Extrapolation of APRStt and DTMF to other local Applications and Events: The full APRStt spec has many formats for position reporting and callsigns, all to fit within the 16 key DTMF memory limit. When data blocks are 4 digits, then full callsigns can be used (10 digits). When up to 7 digits of data are needed (say Marathon runner numbers), then callsign suffixes can be used (5 digits). When up to 9 digits of data are needed, then a 3 digit locally assigned user ID (3 digits) can be used.

Examples of 4 and 6 digit data:

- XXYY gives positions to the nearest mile over a 100 mile square grid area

- XXYY gives positions to the nearest 500' over a 10 mile square grid area

- XXYY gives positions to the nearest 50' in a local event covering a mile (same as GPS)

- XXYY as minutes of Lat/Lon can give 1 mile precision. See Dayton map at right.

- XXXYYY gives positions to the nearest 500' over a 100 mile grid area

- XXXYYY gives positions to the nearest 50' over a ten mile grid area, (GPS resolution)

- NNNNNc can report a runner code "c" for any marathon runner etc
- MMmS can report Status at any tenth of a mile mark in a Marathon
- TTTSSS can report the troop number and score during a jamboree
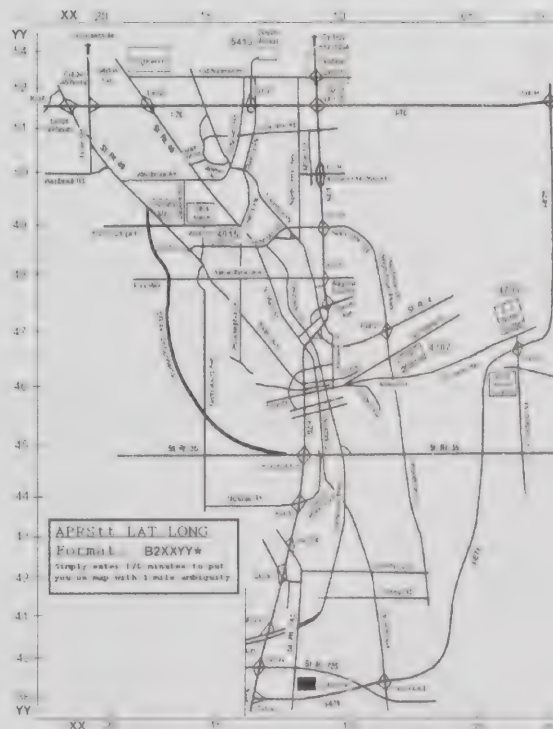
The applications of APRStt are as many and as varied as ham radio itself when there is specific data that needs to be quickly and accurately reported. And any code can be invented for any purpose because all local events are local. Once the code is received at the local APRStt engine, the code is converted to normal APRS packet so that everyone at the event with APRS can see what is going on, and since the APRS-IS then distributes that data worldwide, there is no limit to what can be done.

```
'****************************************************************************|
'***                     PSAT2 list of FLIGHT Messages          18 Nov 2015|
'****************************************************************************|
DTMF messages are sent in 16 key strings starting with the "C", ending in #
BxxMMCCCCCCkkkk# Where CCCCCCkkkk are your callsign keys and key code or
CMMxxCCCCCCkkkk# Where CCCCCCkkkk are your callsign keys and key code and
MM is the message number below and xx is a modifier number spoken in place
of __
Note: Wierd spellings attempt to get the voice on PSAT2 to sound better


'****************************************************************************|
'*                     ARL Emergency Standard Messages          18 Nov 2015|
'****************************************************************************|
 01 Everyone is safe, Do not worry.
 02 I am Coming home as soon as posseble.
 03 In hospetal, Receiving care and recovering.
 04 Only slight property damage here, Do not worry.
 05 I am moving to a new location, Will make contact then.
 06 Will contact you as soon as possible.
 07 Please reply by Amatur Radio.
 08 Need additionel radio equipment for emergency use.
 09 Additionel _ radio operators needed.
 10 Please standby for further information.
 11 Establish Amatur Radio contact on _ meeters.
 12 Ankchus to hear from you.
 13 Medicel emergency sit uation egsits here.
 14 Sit uation here is worsening and becoming criticel.
 15 Please adv eyeze your condition and what help is needed.
 16 Property damage is very significant.
 17 RE ACT communications are on channel _.
 18 Please contact me as soon as posseble.
 19 Request halth and welfare report.
 20 Temporarily stranded, Will need some assistance.
 21 Serch and Rascue assistance is needed.
 22 Need accurate information on conditions at your location.
 23 Report accessebility and best way to reach your location.
 24 Evacuation of razidents from here is urgently needed.
 25 Please adv eyeze weather conditions at your location.
 26 Need help and care for evacuation of sick and injured.
 27 Hi, This was Dove in spaice, anni verse air E
 28 There are _ of us here.
 30 Marytime Emergency Code number _.
 31 We are operating on emergency power.
 32 We are operating on sowlar power.
 33 This is a voice test.
```

```
40 Q S L, your number _, my number is %. (% is the next number 'CQ msg
41 Q S L, your C Q number _. 'CQ msg
42 Q S L, your C Q number _ and thanks for the contact. 'CQ msg
43 Go Navy, beet Army!. '''
44 Navy Beets Army by _. ''
45 I am _ years old.
46 Greetings on your berthday. 'birthday to bertday
47 Got your message number _.
48 I am in school grade _. ''
49 Celebrating _ munths in spaice.
50 Greetings by Amatur Radio. 'Amateur to Amatur
51 Am having a wonderful time.
52 Really enjoyed visiting with you.
53 Received your package, Thank you.
54 Many thanks for your good wishes.
55 Very delighted to hear your good newze.
56 Congratulations on your worthy achievement.
57 Wish we could be twog ether.
58 Have a wonderful time, Let us know when you return.
59 Congratulations on the new arrivel, Hope all are well.
60 Wishing you the best.
61 Wishing you happy holidays and New Year.
62 Greetings and best wishes for the holiday season.
63 Our best wishes are with you, Hope you win.
64 Arrived safely at _ hours.
65 Please meet me on arrival at _ hours.
66 D X Q S Ls are on hand at the Q S L Bureau.
67 Your message _ is undelivereble.
68 Best wishes for a speedy recovery.
69 Welcome, We hope you will enjoy the fun and fellowship.
70 Call me ON my cell at _ Oh clock.
71 No cell phone service here.
72 My Cell phone battery is dead.
73 Greetings from AMSAT, Keeping ham radio in spaice fo _ years.
74 My Cell phone charging opportunitees are limited.
75 Call my cell phone on the hour.
76 My Radio power charging capabilities are limited.
77 My next contact time will be in _ minutes.
78 My next contact time is tomorrow.
79 Please send items number _.
80 I am on schedule.
81 I may be delayed by _ hours.
82 I may be delayed by _ days.
83 I may be earlyer by _ hours.
84 I May be earlyer by _ days.
85 I may quit earlyer by _ stops.
86 I may go further by _ stops.
87 We are camping and enjoying it greatly.
88 Sending love and kisses!.
89 Contact me on the _ meeter band.
90 There are _ of us here.
91 Celebrating _ weeks in spaice.
```

Extending D-STAR with Codec 2

Antony Chazapis, SV9OAN
Heraklion, Crete, Greece
chazapis@gmail.com

Abstract – D-STAR was the first digital voice system designed by radio amateurs specifically for amateur radio use. However, it relies on a proprietary, patented codec, which has been the subject of controversy among amateur radio operators and regulatory authorities. In this paper, we present a protocol extension, that enables the optional use of Codec 2 in voice streams – an open-source and patent-free alternative to the default codec. The "D-STAR vocoder extension" has been implemented in a series of accompanying, open-source software projects, which focus on seamless interoperability with current D-STAR deployments. Transcoding voice streams between codec variants is possible with a custom version of the popular xlxd reflector. In addition, we introduce Estrella, a software-only, radio-like desktop and mobile application, for over-the-network communications.

1. Introduction

Digital Smart Technologies for Amateur Radio (D-STAR) is a digital voice and data standard designed for amateur radio use [1, 2, 3]. The research leading to the original specification [4], published in 2001, was funded by the Japanese government and led by JARL, in a joint effort with Japanese radio equipment manufacturers. D-STAR digital voice transmissions require less bandwidth than analog FM and provision a small percentage of the stream to be used for arbitrary data, like text messages or GPS information. D-STAR repeaters may have modules in several bands (most commonly VHF and UHF) and are capable of forwarding streams from one band to another, or through a special "gateway" module which "links" the repeater to some other Internet-based endpoint. Digital voice, once picked up by a repeater or a "hotspot" (a local, low-power, radio-to-Internet gateway), can be easily relayed over the network to other physical or virtual repeaters (called "reflectors"), allowing users to participate in pre-established or ad hoc worldwide talk groups [5, 6].

However exciting new technologies may be – especially to amateur radio operators, D-STAR was received early on with mixed reviews. Two were the most prominent causes of criticism: the availability of products from a limited set of manufacturers at a relatively high price point, and the use of a closed-source, proprietary codec for compressing raw, digitized voice into a low-bitrate stream and vice versa. Both points are still valid – almost 20 years after the introduction of the standard.

Most of the D-STAR equipment available is manufactured by ICOM, Inc., which almost solely supports and promotes D-STAR technology [7, 8] (actually, "D-STAR" is an ICOM registered trademark). And

while the protocol defines the packaging and modulation details, it does not define the format of additional data inside voice streams, routing protocols, and other relevant extensions that are necessary to support end-to-end D-STAR network deployments and associated applications. ICOM uses proprietary implementations of several D-STAR functions [9, 10], which throughout the years have been reverse-engineered, in order to enable building custom D-STAR radios, repeaters [11], and even faster and more efficient routing mechanisms [12, 13]. The current ability of D-STAR to support such a variety of configurations and transparently bridge to other digital voice systems (like DMR and System Fusion), has been largely driven by a large collection of open-source software and hardware, supported by a vibrant community of amateur radio operators, repeater builders/maintainers, and hardware homebrewers, that continuously devise and implement innovative solutions (a wealth of historical information is included in [14], while early practical examples are shown in [15, 16])

The result of this effort is that a fully operational D-STAR network backbone (repeaters, reflectors, as well as routing and bridging functions) can now be completely implemented using open software, running on low-cost, commodity devices. However, this is not the case with end-user radios. The proprietary codec used by D-STAR, which is available in the form of a hardware chip, renders do-it-yourself radios very difficult to construct. While there have been some software implementations that interface with a chip attached via USB on a remote machine over the network (like DVTool [17] or BlueDV [18]), users mostly prefer the convenience of a self-contained hardware radio – still with a limited choice of options at relatively high price points.

Advanced Multi-Band Excitation (AMBE), the codec used by D-STAR, was initially selected by the designers as the only practical option available at the time. There has been some discussion on what will happen when the respective patent held by Digital Voice Systems, Inc. will expire (if it is actually valid and has not already [19]), however an open implementation would also require a significant amount of information made publicly available by the company, which is most probably unexpected. In practice, variations of the AMBE codec are used in most well-known contemporary digital voice modes, so the codec selection is no longer a source of strong dispute; the proprietary codec issue has even resulted in a government ban of the mode all together in France. An alternative to AMBE, Codec 2, has been available for some time now. Codec 2 is open-source and patent-free, capable of compressing speech to very low bit-rates [20]. Moreover, it has already been used in amateur radio applications, most notably as an integral part of FreeDV, a digital voice mode for HF [21].
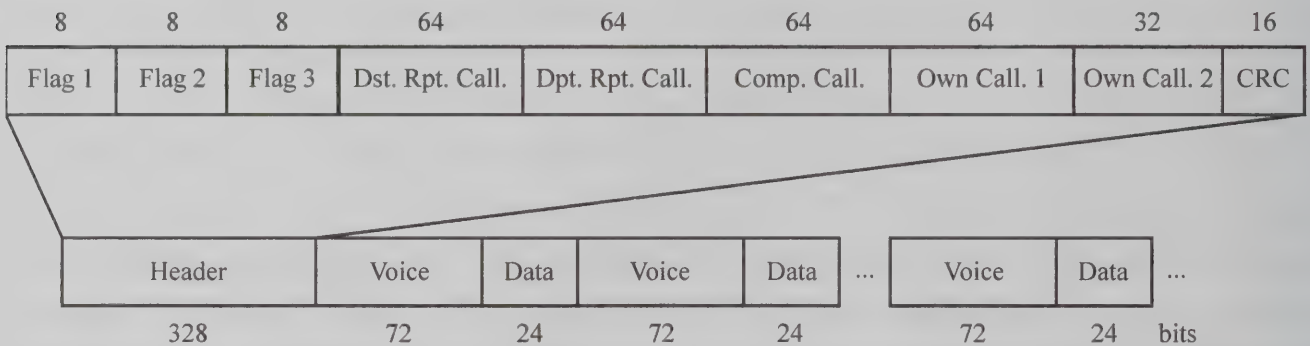
In this paper, we present a protocol extension to enable the optional use of Codec 2 instead of AMBE in D-STAR communications. The possibility of such an implementation had been discussed in the past [22], but – to our knowledge – had never been realized. The open source codec, will hopefully address both aforementioned issues, enabling the emergence of affordable, easy-to-build D-STAR radios, covering a wide spectrum ranging from software-only solutions connecting directly to reflectors, to completely independent hardware devices. Having D-STAR endpoints with no hardware requirements may also

allow an entirely new range of network-based applications, either stand-alone, or as interfaces to existing, external systems.

The changes to the protocol, along with their implications, are discussed in detail in the following chapter. In summary, a currently unused header flag is employed to mark the codec type in voice frames. This work is not meant to replace current D-STAR deployments, but augment them in a fully compatible way. To this end, we have implemented a reflector (based on xlxd [23]) that is capable of transcoding between AMBE and Codec 2 streams. The reflector is part of a series of extension-compatible, open-source software solutions, which also includes a set of command line utilities to experiment with the extension and a software-only client, called "Estrella", available in desktop and mobile versions.

2. Design

The D-STAR protocol specification defines the bit framing used to transmit either data or voice streams. Each stream begins with a synchronization pattern, followed by a header, and – in the case of voice communications – a variable number of alternating voice and data frames (Fig. 1) [24].

8	8	8	64	64	64	64	32	16
Flag 1	Flag 2	Flag 3	Dst. Rpt. Call.	Dpt. Rpt. Call.	Comp. Call.	Own Call. 1	Own Call. 2	CRC

Header	Voice	Data	Voice	Data	...	Voice	Data	...
328	72	24	72	24		72	24	bits

Figure 1: Overview of a D-STAR voice stream

The header, which is 328-bits long carries primarily addressing and routing information: who is sending the stream, where it is headed, through which repeater or gateway, etc. (It is actually coded into 660 bits when transmitted, after bit interleaving and scrambling.) Except callsigns and 2 trailing bytes for the checksum, the header also includes 3 distinct 1-byte long "flags" for identifying the type of communication, triggering control functions, and – what is of particular interest here – protocol expansion. Specifically, "Flag 3" has by default all 8 bits set to zero; any other value is undefined and left for future use.

The 72-bit long voice frames carry the data from the AMBE codec, while the 24-bit long data frames which are interleaved between them, help in synchronization and are commonly used to provide redundant copies of the header, text messages, and other information, like sender location. The AMBE codec encodes speech at 3600 bps, including error-correction information. Thus, each 72-bit voice frame (9 octets) corresponds to exactly 20 ms of voice. No other bits in the D-STAR stream are relevant to voice information.

In the following paragraphs, we will present in detail how the unused bits of Flag 3 can be exploited in order to extend the D-STAR protocol with Codec 2 support. We propose setting specific bits in Flag 3 to signify different content types of the voice frame. Two variants are supported: one with higher voice quality but no forward error correction (FEC), and another one with slightly higher voice compression, but with additional error-correction data piggybacked at the end of each encoded voice chunk.

2.1. D-STAR vocoder extension

The "D-STAR vocoder extension" uses the Flag 3 byte of the header, to mark the vocoder type in the voice frames as follows (in accordance to section 2.1.1, page 4, of the D-STAR specification):

Bit	Meaning	Function
`0000000x`	Vocoder	0: AMBE (backwards compatible) 1: Codec 2
`000000x1`	Mode	0: Codec 2 3200 (160 samples/20 ms into 64 bits) 1: Codec 2 2400 (160 samples/20 ms into 48 bits) plus FEC (22 bits)
`00000100` to `11111111`	Undefined	Use for future expansion

Table 1: Possible values of Flag 3 with the D-STAR vocoder extension

Toggling the least-significant bit of Flag 3 switches between backwards-compatible AMBE mode and the new Codec 2-based voice frames. A second bit controls the bitrate of the codec and the presence of FEC.

Codec 2 has several modes, the highest bitrate one working at 3200 bps – a close match to 3600 bps. Note, however, that the Codec 2 algorithm only provides speech processing and the resulting stream does not include any error-correction. 20 ms of voice in Codec 2 3200 mode require 64 bits (8 octets), which fit in the 72-bit space available, but do not leave much space for implementing FEC. The next available Codec 2 mode, which operates at 2400 bps, requires just 48 bits (6 octets) to encode a speech waveform of equal duration, which leaves an adequate amount of bits free, to protect the first 24 bits of the voice data with two applications of the (23, 12) Golay code. Each application produces an additional 11 bits that are used for error correction at the receiving side, raising the grand total of voice-related bits to 70; 2 short of the space available. (In practice, Codec 2 2400 mode also has 2 spare bits, so the transmission really requires 68 bits.)

This technique of employing FEC, is similar to what is implemented by FreeDV, a digital voice mode built upon Codec 2: FreeDV 1600 mode uses Codec 2 1300 mode to encode 40 ms of speech to 52 bits of voice data and then applies a (23, 12) Golay code to protect a 12-bit selection of those 52 bits. In Codec 2-encoded voice samples, some bits are more "important" than others. In the context of the implementation discussed here, it has been decided to apply the Golay code to the the first 24 bits of the

resulting data chunk encoded with Codec 2 2400 mode, which contains all the voicing and pitch/energy bits, plus the first 14 bits of harmonic magnitudes (Line Spectrum Pairs – LSPs).

2.2. Discussion

The proposed protocol extension is backwards compatible, so traffic using Codec 2 will pass through current D-STAR hardware (repeaters, hotspots, etc.) and software (repeater controllers, reflectors, etc.). Of course, hardware transceivers with AMBE chips are incompatible and cannot be used to directly communicate with Codec 2 extension implementations.

Interoperability between vocoder modes can be established using Internet-based D-STAR reflectors, in a similar fashion that they are currently being widely used to transcode and bridge between D-STAR, DMR, and System Fusion. In particular, in the next chapter we will discuss how xlxd, which is the most widespread software used in this setup, has been extended to allow transparent communications between AMBE-based devices and Codec 2-based, software-only clients. Note that although early tests have confirmed that xlxd is indeed compatible to pass through Codec 2-based transmissions without any changes, we have selected to apply a different connectivity setup for clients that support the extension (on a different UDP port using the familiar DExtra protocol [25], which has been named "DExtra Open"), in order to avoid user confusion and establish a fully compatible path for trouble-free adaptation of the new mode. Codec 2-based clients are thus "isolated" from their AMBE-only counterparts and respective streams are properly transcoded in order to allow cross-codec interaction.

The wide-spread use of Internet-based digital voice, primarily in the form of talking through hotspots, but also via desktop/mobile applications – like DVTool/BlueDV, is the main reason it has been decided to allow selecting between the 2 variants of Codec 2. For over-the-Internet communications, FEC is not really necessary, as malformed UDP packets – however rare – are dropped by the networking layer and never reach the application. It is therefore advised to always use Codec 2 3200 mode in such setups and enjoy the slight increase in voice quality.

In fact, we expect that dropping the AMBE codec from the protocol, hence the requirement of interfacing with a hardware chip, will allow a new generation of software-only clients to be implemented. Such desktop or mobile clients could directly communicate with each other, but to keep up with the usage paradigm of current radio-based transmissions, they will probably connect to some centralized "hub" (a reflector), that will allow broadcasting voice streams to all directly or indirectly linked users. In the next chapter, we also present Estrella, a D-STAR software implementation that allows communicating through existing reflectors without the need of any AMBE hardware.

The open source codec, may also allow home-brewing transceivers using a Rasbperry Pi, and an attached radio, either in the form of a directly connected HAT (Hardware Attached on Top) [26], or an MMDVM modem [27] (even one constructed with through-hole components [28]) interfaced to a "traditional", analog radio. The necessary software could run on the Raspberry Pi itself, assuming a method to

attach a microphone and speaker, or – more likely – on a mobile device running an Estrella variation that instead of transmitting streams directly over the Internet, sends them to the Raspberry Pi over Bluetooth (and vice versa), using an established D-STAR over-the-network protocol like DExtra. Such a setup, which has the user interface separate from the radio instance, may allow a variety of interesting, interoperable implementations, that may in turn cover a multitude of different usability and physical deployment requirements.

3. Implementation

The D-STAR vocoder extension has been implemented in three separate, but interoperable, projects:
- pydv: A Python library and an associated set of executable command-line utilities to interface with D-STAR reflectors and transcode saved voice streams, either locally, or using the transcoding server employed by xlxd (ambed).
- chazapis/xlxd: A "fork" of the leading software used to implement D-STAR reflectors, enhanced with the ability to recognize and transcode Codec 2-based streams to AMBE and vice versa when the appropriate hardware is present. Used as a communications hub for D-STAR software clients implementing the extension and a bridge between such clients and devices (repeaters or hotspots) serving AMBE-only transceivers.
- Estrella: A software-only D-STAR application that implement Codec 2-based communications via a compatible reflector (like the aforementioned xlxd fork). Estrella is available for macOS and iOS, while an Android version is in the works.

All the projects above are open source (licensed under GPL) and freely available at GitHub [29, 30, 31, 32]. Further details for each are presented in the following paragraphs.

3.1. pydv

The pydv project was originally developed as a loose collection of code to assist in the early stages of development of the extension. Now – at version 2.0 – it provides Python interfaces to manage DExtra and DPlus connections, convert from network data to D-STAR streams and vice versa, save and load such streams to/from .dvtool files [33], as well as encode and decode voice data using Codec 2 or AMBE (decode only via mbelib [34]), and transcode via a compatible ambed server (included in chazapis/xlxd).

The following command-line executables are included:
- dv-recorder, which connects to a reflector and records traffic in .dvtool files
- dv-player, which plays back a .dvtool file to a reflector
- dv-encoder, which converts a .wav file to a .dvtool file using the Codec 2 vocoder
- dv-decoder, which converts a .dvtool file using any vocoder to .wav
- dv-transcoder, which connects to an ambed server and converts a .dvtool file using the AMBE vocoder to a .dvtool file using the Codec 2 vocoder and vice versa

These utilities can prove useful to anyone experimenting with D-STAR in general. As a library, pydv may support higher-level applications that require D-STAR reflector connectivity using any of the supported protocols. pydv is written in Python 2.7. Future work includes porting it to Python 3, as well as implementing more digital voice network protocols.

3.2. chazapis/xlxd

xlxd is the most modern D-STAR reflector software. In essence, the function of a reflector is simple: it provides network protocol handlers for clients to connect and submit digital voice streams to specific "modules" (connection contexts, virtual "rooms"). The reflector relays each stream, in real time, to any client connected via any protocol to the same module. Some reflectors can even connect as clients to other reflectors, organizing the network traffic in cross-country or cross-continent meshes, and enabling ad hoc linking of talk groups together based on group-specific policy and habits [35].

There are various already established D-STAR network protocol variants. xlxd implements them all, thus allowing interoperability between them. As such, it can be deployed as either a DPlus, DExtra, or DCS-compatible reflector, using REF, XRF, or DCS callsign prefixes respectively (or all of the above simultaneously). By advertising a reflector installation to the appropriate callsign-to-IP resolution registries, clients can use standard D-STAR commands to establish repeater/hotspot links to the reflector. Moreover, xlxd introduces the notion of cross-reflector "trunks" (using the XLX protocol), which convey the streams of many modules simultaneously, in practice "extending" part of one reflector from one installation to another. A group of linked reflectors is commonly called a "constellation".

Recently, xlxd has also been interfaced to similar DMR network structures, like the BrandMeister network [36]. DMR uses a variant of AMBE to encode/decode speech in transceivers, so the streams running through xlxd in that case are not directly compatible with D-STAR. To cross codec barriers, a secondary software service called ambed, provides transcoding between the two formats. ambed requires the presence of one or more hardware chips to decode data from one AMBE variant into voice and encode it into the other. AMBE chips may have one or more channels; at least two are required to transcode a stream in real time.

Given the success and momentum of xlxd, as well of the eagerness of its developers to support any new and exciting technologies in the digital voice arena, it seemed as the perfect candidate to base the implementation of the D-STAR vocoder extension interoperability with other modes. As discussed, because of the extension's backwards compatibility, xlxd could be deployed as-is to support Codec 2-based clients using any existing D-STAR network protocol, but such an arrangement would magnify incompatibility issues. Instead it was decided to work first – before even starting implementing clients – in the direction of providing a unified substrate for codec-independent communications.

To this end, the xlxd/ambed project was forked and patched as necessary to transcode and bridge voice streams of any existing format to the new extension. The solution implements an additional DExtra lis-

16

tener on a different port (30201 instead of 30001). The new port is to be used by reflectors with the "ORF" callsign prefix (Open ReFlector). Any client connected to an ORF reflector will receive streams encoded with Codec 2. All other protocol handlers will still send out data encoded with AMBE. Note that the protocol/port only affects data transmitted by the reflector. The stream codec is recognized by all protocol handlers, so a client can still transmit data using any codec on any port. The rationale behind this is that DExtra links may be used by repeaters or other reflectors, so it is not really possible to know what their clients support. So, nothing will change when linking a repeater to an XRF reflector, but will do when linking to an ORF one. The new port endpoint has been named "DExtra Open" to distinguish it from the protocol running on the default DExtra port.

A detailed outline of the code changes done is attached to the pull request that has been submitted upstream, so the work will eventually be included into future versions of xlxd. In summary, most of the effort was concentrated into ambed, so each "vocodec" channel will function with three interfaces (1 in/ 2 out, instead of 1 in/1 out). Each such channel now has a (virtual) Codec 2 interface attached, along with the two physical ones for AMBE. When a new incoming stream presents itself, the appropriate transcoding channel is selected depending on the input codec. Channels are grouped together in triplets. Each group contains all possible permutations of respective interfaces – three channels, while only one channel from each group can be used at a given time. Groups represent hardware resources that cannot be shared between streams; channels represent input/output configurations. The rest of the work involved interfacing xlxd to the new "1 codec in, 2 codecs out" ambed interface, the new DExtra Open protocol listener on port 30201, etc. xlxd and ambed are written in C++.

The latest version of chazapis/xlxd is currently running at XLX393/XRF393 [37], which is deployed on a Raspberry Pi and supports transcoding a single stream at a time by employing two DVMEGA DV-stick30 USB interfaces (Figure 2). Operation is not restricted in any way, and it can be freely used for testing and experimenting with new digital voice technologies. XLX393/XRF393 is also reachable via DMR, through TG20209 of the BrandMeister network.

Future work includes deploying an ORF registry to track corresponding reflector deployments. The ORF registry will be useful for Codec 2-based software clients, in order for them to display compatible and available servers for connecting to. Assuming Codec 2-based D-STAR hardware devices at some point, the ORF registry may also tie in to repeater/hotspot software (like the ircDDB Gateway), to enable routing respective linking requests to the appropriate DExtra Open network endpoints.

**Figure 2: XLX393/XRF393 current hardware setup**

3.3. Estrella

Estrella is a radio-like software-only client for DExtra Open reflectors. Two platform variations have been implemented: a desktop-compatible macOS version and a mobile-friendly iOS one. Both use a similar user interface and aim for a very simple user experience: the user enters the connectivity details (callsign and ORF network endpoint), and is then presented with a minimalistic screen showing the connection status and providing a PTT button to initiate transmissions (Figures 3, 4). Although direct client-to-client connections could be possible, it has been decided to only support reflector-based setups, in order to match as close as possible the practice of using a "traditional" RF radio transceiver, encourage the deployment of new reflector software that complies with the proposed vocoder extension, and foster an environment of interoperable cross-codec digital voice applications and devices. As mentioned, with chazapis/xlxd installed and the appropriate hardware, the reflector will handle transcoding and bridging streams of different types and protocols.

Estrella implements the D-STAR vocoder extension and uses a linked Codec 2 library to encode and decode D-STAR communications without the need of any external hardware. macOS and iOS versions are both are written in Objective-C, and share a significant amount of code that is abstracted into 2 libraries (called Frameworks in Objective-C nomenclature) [38, 39]:

• CocoaDV, a Cocoa Framework to manage DExtra connections to D-STAR ORF reflectors
• CocoaCodec2, a Cocoa Framework for the Codec 2 low bit-rate speech codec

CocoaCodec2 bundles the source code of the Codec 2 SVN repository in an Xcode-friendly format. (details on how this was accomplished are given at its GitHub page). CocoaCodec2 is used by CocoaDV,

which in turn provides the necessary network and digital voice stream abstractions to Estrella. The result is that the core Estrella implementation only needs to concentrate on graphical user interface elements, handling connectivity events, stream buffering, and managing the available audio hardware. Actually, as the macOS and iOS core libraries have similar APIs, the respective Estrella codebases are also very similar.
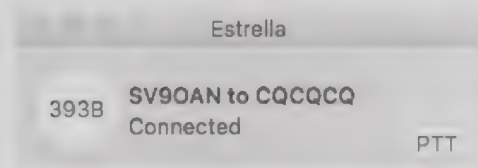
Figure 3: Estrella's main window (macOS version)
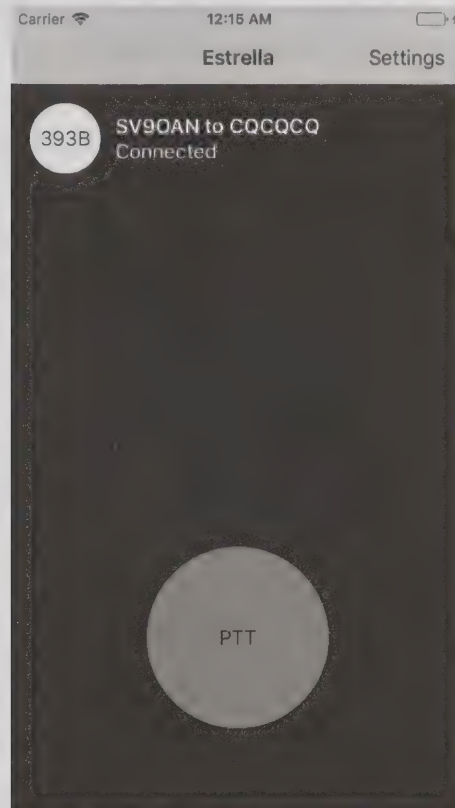
Figure 4: Estrella's main window (iOS version)

Current plans include completing the Android variation of Estrella and releasing all binary forms on corresponding online distribution services (a binary macOS version is already available for download at GitHub). Future versions may present a simpler user interface for configuration, by integrating server selection with the ORF registry, and incorporate text messaging and D-PRS functionality (exchanging positional data and presenting it on a map, as done by APRS [40]).

4. Conclusion

In this paper, we have described the design and implementation of an extension to the D-STAR protocol that uses Codec 2 for voice communications, instead of – or alongside – AMBE. We hope this work to become a significant milestone in D-STAR evolution, as it enables the development of completely open end-user radios and services, seamlessly interoperating with any current deployments. Moreover, the method used here can support other, similar D-STAR extensions, in order to introduce new voice codecs, or other, more radical changes to the stream structure.

In practice, in nearly 20 years of existence, D-STAR has already evolved in many ways. However, the available protocol specification does not include many details that are now considered *de facto*. Back then, it was decided to leave out some implementation specifics: the structure of text messages or location information included in voice streams, the routing techniques and associated callsign discovery methods, the format of streams when relayed over the network, etc. Many of these functions have been implemented in proprietary extensions, that – although mostly reverse-engineered – are for obvious reasons poorly documented. Discovering how D-STAR radios, repeaters, gateways, and reflectors work is a formidable task, as anyone interested can only find information in scarce Internet resources – that may vanish at any point – or deep into the codebase of related software projects. This unfortunate fact, neither matches the open nature of amateur radio, nor aligns to the (presumably) original intention of the D-STAR design group: to publish an open document that will provide the basis of cooperation and interoperability between radio amateurs and equipment manufacturing companies.

Adoption of digital radio technologies is growing larger every year, so D-STAR is now more relevant than ever. Both as a standard for digital radio transmissions, and as the foundation of an Internet-based, worldwide communication network for amateur radio. Sooner rather than later, JARL – or any other big and respectable amateur radio association – should convene a new standards body to release a revised, modern D-STAR standard, documenting established usage of the protocol and providing new directions for the future. If that ever happens, hopefully an open voice codec will indeed be an option – either the one presented here, or any other patent-free solution; in the true spirit of amateur radio.

Notes – *D-STAR is a registered trademark of ICOM, Inc. AMBE is a registered trademark of Digital Voice Systems, Inc.*

References

[1] John Gibbs, KC7YXD, "D-STAR: New Modes for VHF/UHF Amateur Radio, Part 1", *QEX*, Jul/Aug 2003, p. 30–34. http://www.icomamerica.com/en/downloads/DownloadDocument.aspx?Document=604

[2] John Gibbs, KC7YXD, "D-STAR, Part 2: Design Considerations", *QEX*, Sept/Oct 2003, p. 22–28. http://www.icomamerica.com/en/downloads/DownloadDocument.aspx?Document=605

[3] John Gibbs, KC7YXD, "D-STAR, Part 3: Implementation", *QEX*, Noc/Dec 2003, p. 42–47. http://www.icomamerica.com/en/downloads/DownloadDocument.aspx?Document=606

[4] "D-STAR System". https://www.jarl.com/d-star/shogen.pdf

[5] Toshen M. Golias, KEØFHS, "Amateur Radio Notes: Diving into D-STAR", https://amateurradionotes.com/d-star.htm

[6] Toshen M. Golias, KEØFHS, "Amateur Radio Notes: Hanging out with hotspots", https://amateurradionotes.com/hotspots.htm

[7] ICOM, Inc., "D-STAR System Introduction". http://www.icomamerica.com/en/downloads/DownloadDocument.aspx?Document=447

[8] ICOM, Inc., "D-STAR. For the Second Century of Amateur Radio". http://www.icomamerica.com/en/downloads/DownloadDocument.aspx?Document=366

[9] Peter Loveall, AE5P, "D-STAR Uncovered", *Proceedings of the 27th ARRL and TAPR Digital Communications Conference*, Sept 2008. https://www.tapr.org/pdf/DCC2008-D-STAR-AE5PL.pdf

[10] Jonathan Naylor, G4KLX, "The Format of D-Star Slow Data". https://www.qsl.net/kb9mwr/projects/dv/dstar/Slow%20Data.pdf

[11] Jonathan Naylor, G4KLX, "g4klx/DStarRepeater: The D-Star Repeater.", https://github.com/g4klx/DStarRepeater

[12] "ircDDB". http://db0fhn.efi.fh-nuernberg.de/doku.php?id=projects:dstar:ircddb

[13] Jonathan Naylor, G4KLX, "g4klx/ircDDBGateway: The ircDDB Gateway for D-Star", https://github.com/g4klx/ircDDBGateway

[14] Steve Lampereur, KB9MWR, "Digital Voice applications and Ham Radio". https://www.qsl.net/kb9mwr/projects/dv/plan.html

[15] John Hays, K7VE, "D-STAR for the Technically Curious", *Presentation at the 29th ARRL and TAPR Digital Communications Conference*, Sept 2010. https://www.tapr.org/pdf/DCC2010-D-STAR-technical-K7VE.pdf

[16] Mark Braunstein, WA4KFZ, "Introduction to D-STAR Basics", *Presentation at the 30th ARRL and TAPR Digital Communications Conference*, Sept 2011. http://www.tapr.net/pdf/DCC2011-Intro_to_D-Star-Mark_Braunstein_WA4KFZ.pdf

[17] "DV Dongle". http://www.dvdongle.com

[18] "BlueDV". https://www.pa7lim.nl/bluedv/

[19] Bruce Perens, K6BP, "AMBE Exposed", *Presentation at the 33rd ARRL and TAPR Digital Communications Conference*, Sept 2014. https://www.qsl.net/kb9mwr/projects/dv/codec/AMBE_Exposed.pdf

[20] David Rowe, VK5DGR, "Codec 2 - Open Source Speech Coding at 2400 bit/s and Below", *Proceedings of the 30th ARRL and TAPR Digital Communications Conference*, Sept 2011. http://www.tapr.net/pdf/DCC2011-Codec2-VK5DGR.pdf

[21] "FreeDV: Open Source Amateur Radio Voice". https://freedv.org

[22] Bruce Perens, K6BP, and David Rowe, VK5DGR, "Codec2: An Open Future for Digital Voice", *Presentation at the 29th ARRL and TAPR Digital Communications Conference*, Sept 2010. https://www.tapr.org/pdf/DCC2010-Codec2-presentation-K6BP-VK5DGR.odp

[23] Jean-Luc Deltombe, LX3JL, "LX3JL/xlxd: HAM radio multiprotocol dstar reflector server". https://github.com/LX3JL/xlxd

[24] Denis Bederov, DL3OCK, "DSTAR radio frame structure in DV mode". http://db0fhn.efi.fh-nuernberg.de/lib/exe/fetch.php?media=projects:dstar:ircddb:dstar_dv_frame3_en.pdf

[25] Steve Lampereur, KB9MWR, "D-Star Open Source / Dextra project by Scott Lawson, KI4LKF". https://www.qsl.net/kb9mwr/projects/dv/ki4lkf/ki4lkf.html

[26] Mathis Schmieder, DB9MAT, "mathisschmieder/MMDVM_HS_Hat: MMDVM_HS Hat for the Raspberry Pi (Zero)". https://github.com/mathisschmieder/MMDVM_HS_Hat

[27] Jonathan Naylor, G4KLX, "g4klx/MMDVM: The firmware for the MMDVM (Multi-Mode Digital Voice Modem)". https://github.com/g4klx/MMDVM

[28] Florian Wolters, DF2ET, "Handcrafted MMDVM Adapter", https://www.florian-wolters.de/blog/2016/02/25/handcrafted-mmdvm-adapter/

[29] Antony Chazapis, SV9OAN, "chazapis/pydv: D-STAR library and utilities in Python". https://github.com/chazapis/pydv

[30] Antony Chazapis, SV9OAN, "chazapis/xlxd: HAM radio multiprotocol dstar reflector server". https://github.com/chazapis/xlxd

[31] Antony Chazapis, SV9OAN, "chazapis/Estrella-macOS: ORF reflector client for macOS". https://github.com/chazapis/Estrella-macOS

[32] Antony Chazapis, SV9OAN, "chazapis/Estrella-iOS: ORF reflector client for iOS". https://github.com/chazapis/Estrella-iOS

[33] Kristoff Bonne, ON1ARF, "Format of files and UDP-streams used on D-STAR". https://qsl.net/kb9mwr/projects/dv/dstar/formats%20of%20files%20and%20UDP-streams%20used%20on%20D-STAR.pdf

[34] "szechyjs/mbelib: P25 Phase 1 and ProVoice vocoder". https://github.com/szechyjs/mbelib

[35] Bob Scott, W6KD, "Reflections on Reflectors". https://qsl.net/kb9mwr/projects/dv/dstar/Reflections%20on%20Reflectors%201.02.pdf

[36] "BrandMeister". https://brandmeister.network

[37] "XLX393 Multiprotocol Gateway". http://reflector.rasc.gr

[38] Antony Chazapis, SV9OAN, "chazapis/CocoaDV: Cocoa Framework for D-STAR". https://github.com/chazapis/CocoaDV

[39] Antony Chazapis, SV9OAN, "chazapis/CocoaCodec2: Cocoa Framework for Codec 2". https://github.com/chazapis/CocoaCodec2

[40] Peter Loveall, AE5PL, "APRS and D-STAR = D-PRS", *Proceedings of the 26th ARRL and TAPR Digital Communications Conference*, Sept 2007. https://www.tapr.org/pdf/DCC2007-D-PRS-AE5PL.pdf

About the author

Antony Chazapis, SV9OAN, is licensed since 2009. He is a member of RAAG, RASC, ARRL, and the volunteer group HARES (Hellenic Amateur Radio Emergency Service). Since 2017, he has been serving as the Association Manager for the SOTA award program in Greece. Antony enjoys DXing, contesting and exploring what makes digital modes tick. He is the author of PocketPacket, an APRS client application for macOS and iOS. His work on D-STAR includes Estrella, an open-source, software-only radio for macOS and iOS, and associated server software and utilities, that enable D-STAR communications using Codec 2. He holds a PhD in Computer Science and is currently employed by a Toronto-based tech startup implementing bleeding-edge storage solutions.

IPV6 FOR AMATEUR RADIO

DANIEL ESTÉVEZ, EA4GPZ / M0HXM

ABSTRACT. Amateur radio operators have allocated the IPv4 AMPRNet block 44.0.0.0/8. This is routinely used to support several operational networks and experiments, such as the Hamnet. With the increasing exhaustion of IPv4 address space and the goal of using and advancing state-of-the-art technology, it seems appropriate to start using IPv6 for this kind of Amateur activities. This paper gives a proposal of how to distribute IPv6 globally routable address space for Amateur radio use. We also explain some of the advantages of using IPv6, in comparison to the current IPv4 scheme.

1. INTRODUCTION

Currently, Amateur radio operators have the IPv4 AMPRNet [1] block 44.0.0.0/8 allocated for use in all kinds of Amateur radio related activities, such as organizing computer networks which run partially or completely over radio frequency links, and partially over the internet, or doing research and experimentation related to TCP/IP networking. An example of a large network running over the AMPRNet allocation is Hamnet [2].

This large /8 block of IPv4 address space represents more than enough to cover the necessities of all the projects that have emerged, both now and for the forseable future. However, managing and using this IPv4 address space is not exempt of inconveniences, which in the author's opinion could be solved by using IPv6 addressing instead.

First of all, Amateur radio operators are required by the ITU Radio Regulations to identify their own transmissions with their callsign. An IPv4 network for Amateur radio should have a way to map IPv4 addresses to Amateur radio callsigns, since packets can eventually be routed through radio frequency links. However, IPv4 does not provide a good solution to this problem.

Currently, this is solved in an ad-hoc manner in most cases. The endpoints of a radio frequency link comply with the regulations by broadcasting their own callsign periodically, often piggybacking this information onto an existing discovery protocol such as CDP [3]. While this may seem enough to comply with the regulations, it is often impossible to identify which callsign originated a packet that is being routed through a radio frequency link. The only data at hand is the IPv4 source address, and it is only possible to map it into a callsign using an online database such as HamnetDB [4] in limited circumstances.

On the other hand, IPv6 provides an elegant way of mapping addresses onto Amateur callsigns. The 128 bit address length of IPv6 is so large than in many applications the network part of the address uses only 64 bits, while the remaining 64 least significant bits are left to identify the device. This freedom to construct an IPv6 address is used in many applications. An example of this is the SLAAC protocol [5], which allows devices to construct an IPv6 address using their 48 bit MAC address, simplifying IPv6 address configuration. The same freedom can be used both to construct an IPv6 address from an Amateur radio callsign and to map an IPv6 address following this construction to an Amateur radio callsign.

The second disadvantage is centralized management. Even though the 44.0.0.0/8 block is very large, as any shared resource, allocations of this block need to be managed centrally to prevent addressing conflicts. Currently, AMPRNet hands off large sub-blocks to countries, which in turn split their sub-blocks into projects or individuals. All this management is a time consuming process and is prone to disputes.

Here, the advantage of IPv6 is that the address space is so large that it is relatively easy for an individual to obtain a large block of address space for his own private and/or Amateur radio use. Many ISPs are giving off /48 or /56 blocks to individual customers (from which 2^{16} or 2^8 different /64 networks can be extracted, respectively), and it is also possible to obtain similarly large networks from tunnel providers such as Hurricane Electric [6] for individuals not having a native IPv6 ISP. This

makes it possible to construct a decentralized network where individuals or projects use their own IPv6 address space, rather than having to obtain it from a common pool allocated for Amateur radio. Thus, management costs and possible conflicts are reduced significantly.

Finally, another disadvantage of IPv4 is address exhaustion. While it seems that the Amateur radio community will manage to maintain the allocation of the 44.0.0.0/8 block for the near future, IPv4 addresses are by now a very scarce resource, and this large block represents a huge commercial interest. Several large organizations have relinquished unused address space that was allocated to them in the early times of the internet. Therefore, it is not impossible that the Amateur radio community might be forced to free out some of their IPv4 space.

Motivated by how IPv6 manages to solve these problems, and in the interest of advancing Amateur radio technology to the state-of-the-art by introducing IPv6 into our networks and experiments, this short paper gives a concrete proposal about how to manage the addressing of an IPv6 Amateur radio network.

The remaining part of this paper is organized as follows. In Section 2 we show how to encode Amateur radio callsigns into IPv6 addresses. Section 3 gives a description of the proposed IPv6 Amateur radio network, as well as it advantages in comparison to the current AMPRNet. An example of the configuration of a network site using the ideas given here appears in Section 4. Section 5 gives some instructions for people interested in joining the network. Some open research ideas are given in Section 6.

The paper is based on a text hosted in the author's blog [7], which in turn was based on some posts of the author in the 44net mailing list around 2016 outlining the proposal given here.

2. Encoding Amateur callsigns into IPv6 addresses

As mentioned in the Introduction, one of the key ideas of the proposal described in this article is that all the IPv6 addresses used for Amateur radio should encode a valid Amateur radio callsign. In this section we show how to do this.

Amateur radio callsigns are usually composed of up to 6 alphanumeric characters, so it is apparent that there is plenty of space to encode an Amateur callsign into the 64 least significant bits of an IPv6 address. There are several proposals giving concrete methods of how to do this [8, 9, 10]. In this paper, we consider [8], by Robert S. Quattlebaum, which seems the more complete proposal. However, the author believes that an Amateur radio IPv6 network should be agnostic to the method used to encode callsigns into IPv6 addresses. Each participating subnet should be free to choose its own encoding method, provided that this is consistently used and registered in some online database.

For the reader's convenience, let us briefly mention the most important features of the encoding method given in [8]. Amateur callsigns are first encoded in a BASE40 character set, which comprises alphanumeric characters, as well as '-' and '/'. This allows the formation of more complicated callsigns, such as EA4GPZ/P, EA/M0HXM, and also callsigns akin to the SSIDs used in AX.25 [11] to identify different equipment belonging to the same operator (for instance, EA4GPZ-7).

A so called *chunk encoding* is given to encode three BASE40 characters into 16 bits. This makes it possible to encode up to 12 characters in 64 bits, thus allowing most kinds of non-standard callsigns. There are certain advantages to using callsigns shorter than this maximum. For instance, all callsigns up to 8 characters long (and a few 9 character callsigns) can be encoded into a valid EUI-48 address. EUI-48 addresses are used as MAC addresses in several link layer networks, such as Ethernet. The possibility to encode Amateur callsigns into not only IPv6 addresses but also Ethernet addresses allows us to have identification directly at the link layer level and also to derive IPv6 addresses automatically by using SLAAC.

3. The IPv6 Amateur radio network and its advantages

3.1. A network of whitelisted subnets. Another important idea of this proposal is to try to eliminate centralization and management by requiring individuals and projects to obtain address space by their own means rather than taking it from a common pool allocated to the Amateur radio community. With the very large address space of IPv6, and given its current and foreseeable use, this is much more feasible than in IPv4. Note that this view is contrary to some ongoing developments in the Amateur radio community. For instance, the IARU R1 VHF Handbook [12] recommends that a global IPv6 allocation be obtained for Amateur radio usage. In the opinion of the author, requiring that each participant uses their own address space not only simplifies administrative costs, but also has some associated technical advantages, such as solving the problems with the source address filtering done by most ISPs.

Since the IPv6 Amateur radio network is not envisioned as any particular subnet of the globally routable IPv6 address space, there must be something that gives consistence to the network. The idea is that the network is formed by an aggregation of unrelated IPv6 globally routable subnets such that:

- The packets that originate from these subnets are valid for routing via an Amateur radio frequency link. In particular, the traffic originates from a duly licenced Amateur radio operator.
- Each of these subnets is using a certain mechanism to encode Amateur callsigns into IPv6 addresses as a means to identify the Amateur radio operator which originated the traffic.

Therefore, the element that gives consistence to the IPv6 Amateur radio network is a database of the participating subnets. This database is intended to be used as a kind of whitelist when determining if a certain IPv6 packet is fit for being routed through a radio frequency link, because both the source and destination addresses belong to one of the subnets in the whitelist. The database is quite simple. It only contains an entry for each subnet, with some metadata, such as contact details for the person in charge of the subnet, and also the encoding method that each particular subnet is using to encode Amateur callsigns into IPv6 addresses. In this way, anyone can find the callsign responsible for a particular IPv6 address in the network, by first looking up the subnet in the database and taking note of the encoding method, and then using it to compute the callsign.

By now, it is not so clear how to publish and maintain this database of whitelisted subnets. It seems that an online platform such as a wiki or custom web application could be appropriate. The author has also made some experiments by publishing the whitelisted networks using BGP.

3.2. **Reduced administration costs.** The maintenance of a simple database of subnets participating in the Amateur radio IPv6 network is a much simpler task than the allocation of the shared 44.0.0.0/8 block to interested individuals. Therefore, it seems likely that this method of constructing and administrating the network will reduce both the management effort and the conflicts significantly.

In the same spirit of "every individual contributes his own resources to the network", it is also envisioned that each participating subnet or individual manages DNS by his own means. Currently, the domain `ampr.org` is used for most AMPRNet related projects and applications. This also represents an added administration effort. Reverse DNS for AMPRNet addresses is handled by the same means as the `ampr.org` domain. It seems more appropriate that each individual or project manages DNS as deemed convenient, and obtains delegations for the reverse DNS from their ISP in the appropriate manner.

3.3. **Identification of Amateur traffic.** According to the ITU Radio Regulations as well as the Amateur radio regulations in many countries, all Amateur transmissions must be identified with the callsign of the transmitting station. Moreover, there are other limitations regarding transmission of information by Amateur radio means. For instance, most countries explicitly forbid *third party traffic*, which is understood as the retransmission of messages not originated by or not intended for a licenced Amateur radio operator. Since the Amateur service cannot be used with any pecuniary interest, the contents of the allowable messages are also somewhat limited in this respect. Finally, encryption of messages is forbidden both by the ITU Radio Regulations and by many countries (except in special cases of emergency). Encryption is so often used in networking protocols that some care needs to be taken to decide which network traffic is fit for routing over an Amateur radio frequency link.

Often, this problem is not solved properly. Little is done in many cases, by using ad-hoc solutions that have evident flaws. While it does not seem easy to develop an approach that is completely fool proof, IPv6 seems to offer some technical mechanisms that can help. First of all, the requirement that all the IPv6 addresses participating in the Amateur radio IPv6 network embed the callsign of the operator responsible for that device and transmission serves both to satisfy the requirement of identification of Amateur transmissions and to prevent third party traffic. By making sure that a certain IPv6 packet has a source and destination address belonging to Amateur operators, we can be sure that its payload is not to be considered as third party traffic.

Clearly, some degree of trust is put on the people participating on the network. This scheme only works well if IPv6 addresses belonging to the Amateur radio network are used solely for Amateur radio purposes (i.e., for transmissions acceptable for the Amateur service, in particular, not having any pecuniary interest or encryption). In any case, the addressing scheme also serves as a way of separating Amateur and non-Amateur equipment in the network of an Amateur operator. Both kinds of equipment often share a home network. By dividing such network into two subnets and only publishing one in the Amateur radio whitelist, the distinction between non-Amateur and Amateur traffic is simplified.

Additionally, the whitelist can also be used to distinguish which online services can be accessed only by licenced Amateur operators. For many online applications, in particular those that may cause

radio frequency transmissions, such as joining a repeater voice-over-IP conference or operating a club transceiver remotely, it is necessary to identify the user as a licenced Amateur operator. There is no readily available solution for doing this, and each service resorts to its own method, which usually involves maintaining a database of logins for allowed users.

On the other hand, there are also many uses that can be offered to the wide public over the internet, both as a way of showing Amateur radio publicly and for a matter of convenience. Some examples of these are WebSDRs or listening to a repeater voice-over-IP conference (without transmitting).

Services of these two types are often mixed and it is an added burden to separate which parts need authentication and which do not, besides implementing an appropriate authentication method. The IPv6 Amateur radio network can simplify this division. Services that require a licenced Amateur operator can be configured to be accessible only from whitelisted IPv6 subnets belonging to the network.

3.4. Source address filtering. Another technical problem that affects the current AMPRNet is that of source address filtering. Most ISPs will drop uplink packets that are sent from a source address other than the IP address allocated to the user by the ISP. This prevents users from using addresses from the 44.0.0.0/8 block directly on the internet, since the outgoing packets would be blocked at their ISP.

The usual solution to this problem is to use IPv4-in-IPv4 tunnels (or other tunnelling protocols) to interconnect the different AMPRNet stations over the network. However, this makes most of the AMPRNet seem like a large VPN and makes the use of globally routable IP addresses less useful. A large VPN would also work well with private IP addresses. Strictly speaking, only the few stations or subnets that are announcing their AMPRNet addresses by BGP on the internet are really using the 44.0.0.0/8 block over the internet.

The IPv6 Amateur radio network solves this problem because every user is required to use their own address space, obtained from their ISP or a tunnel provider. In this way, such an address space can be used over the internet with no limitations. Subnets participating in the Amateur network can be connected directly over the internet, without the need for any tunnelling, and IPv6 addresses participating in the network can also be used to communicate with non-Amateur equipment on the internet (often for purposes as important as downloading software updates).

3.5. Large blocks for National Societies. Though the proposal given in this paper envisions that each individual or small project would be required to obtain their own IPv6 address space directly from their ISP (or a tunnel provider), it also allows for the possibility that large National Societies or other Amateur societies, such as the IARU regions, may be able to request a larger block directly from their corresponding RIR, announce that block by BGP directly on the internet and use it to give access by radio frequency or other means to interested Amateur operators. In this way, such a society would effectively become an ISP. This approach could still find some problems regarding source address filtering, as detailed above, but nevertheless is compatible with all the ideas outlined in this paper.

4. A CASE STUDY: THE NETWORK AT EA4GPZ

In this section we give a concrete example by showing how some of the Amateur radio equipment in the home network of the author has been configured following the ideas in this paper.

The block that EA4GPZ is using is 2001:470:6915:8000::/49, which has been obtained from the IPv6 tunnel broker at Hurrican Electric [6]. The method used to encode Amateur callsigns into addresses is the BASE40 method described above.

There are the following devices and callsigns in the network:

- Router: EA4GPZ-X
- Server: EA4GPZ-Z
- User access, 2.4GHz: EA4GPZ-S
- User access, 5GHz: EA4GPZ-C

Note: EA4GPZ-S and EA4GPZ-C are currently offline, but the IPv6 addresses are assigned in the DNS.

The callsigns -X and -Z have been chosen because they somehow seem suggesting for a router and a server. The callsigns -S and -C refer to S band (2.4GHz) and C band (5GHz).

The following EUI-48 addresses are assigned to the callsigns above, using the method given in this paper:

- EA4GPZ-X 42:1F:87:2E:5A:F1
- EA4GPZ-Z 92:1F:87:2E:5A:F1

- EA4GPZ-S 7A:1F:87:2E:5A:F0
- EA4GPZ-C FA:1F:87:2E:5A:ED

These EUI-48 addresses can be used directly as Ethernet MAC addresses, which is what is being currently done.

Following the Hamnet terminology, the network 2001:470:6915:8000::/64 is used as a Service-Network and the network 2001:470:6915:8001::/64 is used as a User-Network. This means that servers run in the Service-Network and users that connect by the radio frequency access points get addresses from the User-Network.

The router has the following statically assigned IPv6 addresses:

- 2001:470:6915:8000:421f:87ff:fe2e:5af1
- 2001:470:6915:8001:421f:87ff:fe2e:5af1

In this way, the following IPv6 addresses are generated automatically using SLAAC:

- EA4GPZ-Z 2001:470:6915:8000:901f:87ff:fe2e:5af1
- EA4GPZ-S 2001:470:6915:8001:781f:87ff:fe2e:5af0
- EA4GPZ-C 2001:470:6915:8001:f81f:87ff:fe2e:5aed

The devices which connect through the radio frequency access and have their MAC address properly set up with their callsign using a BASE40-derived EUI-48 address will also obtain, by using SLAAC, an IPv6 address that encodes their callsign.

The IPv6 addresses listed above are published in DNS and reverse DNS using the names:

- EA4GPZ-X router.ea4gpz.destevez.net
- EA4GPZ-Z ea4gpz.destevez.net
- EA4GPZ-S user-2ghz.ea4gpz.destevez.net
- EA4GPZ-C user-5ghz.ea4gpz.destevez.net

Ping from the internet is allowed to all these IP addresses. Access from the internet to the following services running in ea4gpz.destevez.net is also allowed:

- ssh
- mumble
- dxspider (port 7300)

5. JOINING THE IPv6 AMATEUR RADIO NETWORK

The author believes that a key to the success of a good idea is that it gains popularity and adoption. So far, not many people have become interested about an IPv6 Amateur radio network as detailed in this paper. There can be several reasons for this, such as the limited number of Amateur operators interested in this kind of technologies, a limited time for the hobby that gets spent on other aspects of it, or simply that not enough publicity has been given to these ideas (hopefully this paper will try to solve this last cause).

Interested Amateur radio operators are invited to contact the author at the email address given at the end of the paper. As the IPv6 Amateur radio network is a decentralized network where each user offers his own network resources, it is up to each individual to join it. Interested people with some knowledge of IPv6 are already able to join the network. This section gives an overview of the steps needed to do so, depending on the internet connection of the user.

People which already have access to the IPv6 internet and have a globally routable block which is larger than a /64 and that they can administer can use some subnet of this block for their Amateur radio equipment. Probably the best idea is to use a /64 as a Service-Network and another /64 as User-Network if user access by radio frequency is to be expected. It is recommended to allocate a subnet larger than these two /64s, as it may become necessary to give blocks to stations connected directly by radio frequency in the future. An idea is to allocate half of the block available from the ISP for personal or home use and the other half for Amateur radio use.

Then it is necessary to chose a method to encode callsigns into IPv6 addresses. The method given in [8] is recommended. It is mandatory that all the globally routable IPv6 addresses in use have a valid callsign associated using the chosen method. It is also recommended to do so for link-local addresses, but it is not necessary.

People which do not have access to the IPv6 internet but have a static IPv4 address can obtain a /48 IPv6 subnet by at least two methods:

- Using 6to4. This only requires some setup on the router. It also requires access to a 6to4 relay (the anycasted IPv4 address 192.88.99.1). Some ISPs do not give access to a relay. The performance depends on the 6to4 relay you access (there is no control about it, as this depends depends on the ISP's routing). The /48 block obtained depends only on the static IPv4 address.
- Using a tunnel broker such as Hurricane Electric [6]. This requires a registration in the broker's webpage and the set up of the tunnel, as well as some configuration in the router. The performance depends on the other end of the tunnel. The /48 block obtained in this manner is assigned by the tunnel broker from his own network.

Depending on the ISP, these methods could give better or worst performance regarding latency and bandwidth. It is possible to try out both methods and choose the one that gives better performance.

People which have a dynamic IPv4 address can use the tunnel broker from Hurricane Electric [6]. It offers a service to update the tunnel's IPv4 address, which is compatible with dyndns' update protocol.

6. RESEARCH IDEAS

Here we give some ideas which seem interesting topics for research or experimentation, thus advancing the state-of-the-art of Amateur radio.

- Mobile IP. The author has already performed some preliminary tests with UMIP [13].
- Using Differentiated Services to decide if the traffic should be routed by radio frequency links or through the internet (whenever both alternatives are possible). This solves the existing problem that many sites are connected both by the radio frequency links and by the internet, and it is never clear which to prefer when routing the traffic, since certain routes would be best depending on the intended application.
- Using NAT64, CLAT or SIIT to allow access to the AMPRNet from the Amateur radio IPv6 network.
- Using WHOIS to store the contact data for each subnet participating in the Amateur radio IPv6 network.

REFERENCES

[1] Amateur Radio Digital Communications, Managing the AMPRNet – TCP/IP Networking for Amateur Radio, https://www.ampr.org/
[2] Highspeed Amateurradio Multimedia NETwork, https://hamnet.eu/
[3] Wikipedia, Cisco Discovery Protocol, https://en.wikipedia.org/wiki/Cisco_Discovery_Protocol
[4] HamnetDB, Hamnet IP-Database http://hamnetdb.net/
[5] RFC4862: IPv6 Stateless Address Autoconfiguration, https://tools.ietf.org/html/rfc4862
[6] Hurricane Electric Free IPv6 Tunnel Broker https://tunnelbroker.net/
[7] Daniel Estévez, IPv6 for Amateur radio https://destevez.net/ipv6-for-amateur-radio/
[8] Robert S. Quattlebaum, Amateur Radio Numeric Callsign Encoding, https://github.com/darconeous/ham-addr/blob/master/n6drc-arnce.md
[9] ham-ipv6 https://sourceforge.net/projects/hamv6/
[10] Matti Aarnio, Some idea for IPv6 addressing for radio-amateur nodes, http://ham.zmailer.org/oh2mqk/packet-ipv6.html
[11] TAPR, AX.25 Link Access Protocol for Amateur Packet Radio, https://www.tapr.org/pub_ax25.html
[12] IARU R1, VHF Handbook V8.12 https://www.iaru-r1.org/index.php/downloads/func-startdown/1009/
[13] UMIP, Mobile IPv6 and NEMO for Linux, http://www.umip.org/

E-mail address: daniel@destevez.net

Synchronization in FT8

Mike Hasselbeck, WB2FKO

mph@sportscliche.com

Abstract

Deconstruction of the FT8 open-source *FORTRAN* code provides insight on the synchronization algorithm. The heart of the synchronization scheme is a Costas Array, a specially designed square matrix that was invented in 1965 to improve the reliability of underwater sonar. An intuitive explanation of the Costas Array is given, followed by a detailed description of its implementation in the FT8 decoder.

Keywords: FT8, WSJT-X, Costas Array

Introduction. FT8 is a sub-mode of WSJT-X that has become extremely popular for working DX because it enables fast, efficient communication in marginal, weak signal conditions. Forward Error Correction makes this possible, but it requires that transmitting stations and receiver be synchronized to better than 20 ms in time and less than 1 Hz in frequency. Such precision is generally not possible with amateur radio equipment using external reference clocks, so the protocol must supply its own synchronization signal. The different WSJT-X modes accomplish this in a variety of ways, depending on the design requirements. FT8 uses a 7x7 Costas Array [1]. The FT8 decoder performs a coarse search for Costas synchronization symbols in the 15-second frequency waterfall. Synchronization signals that are above a defined energy threshold are identified as candidates for additional decoding. Candidates are adjusted to have initial time and frequency offset accuracy of about 40 ms and 3 Hz. The synch signal of each candidate is then used in a second calculation that performs a quasi-coherent cross-correlation to fine-tune the time and frequency alignment between one or more transmitting stations and the reference frame of the decoder.

Principle of the Costas Array. The following pictorial example can provide an intuitive understanding. An example of a non-Costas array is the diagonal 3x3 matrix M. All the matrix elements are zero except along the diagonal at positions $M_{1,1}$, $M_{2,2}$, and $M_{3,3}$. Render the matrix as a mask, with transparent holes at the three diagonal positions (Figure 1, left). The mask is the frame of reference. An identical signal array is configured with LEDs at the same positions as the mask holes (Figure 1, right). Each LED contributes 1 unit of signal. To get the maximum light transmission of 3 through the mask, it must be perfectly aligned with the LED array.
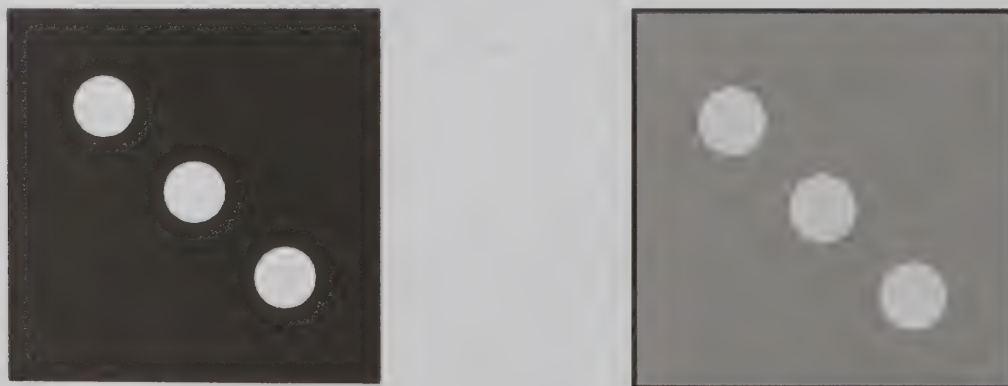

Figure 1: 3x3 diagonal mask (left) and LED array (right).
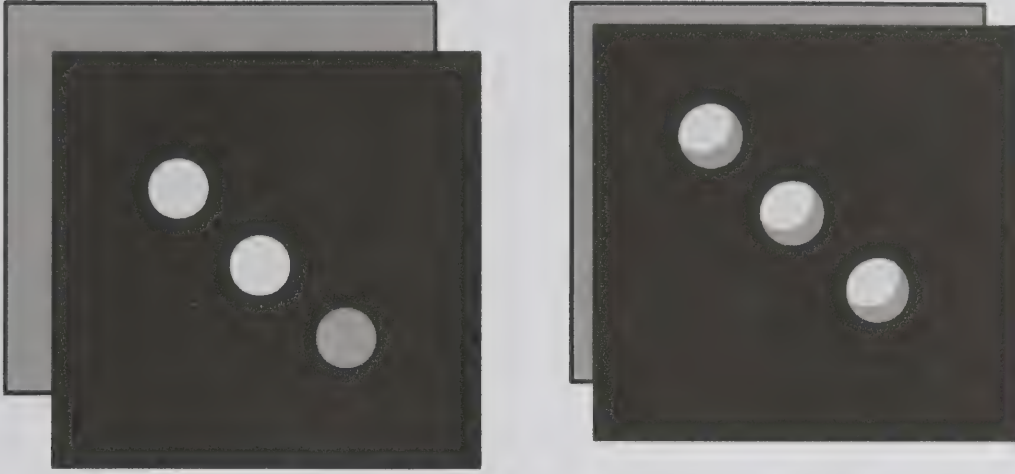
Figure 2: Misalignment of the mask by 1 array element on each axis (left); Transmission approaching maximum (right).

A systematic search of x and y is used to find the maximum brightness. Misalignment by an amount $\Delta x = -1$ and $\Delta y = +1$ is shown in Figure 2, left. Incremental, trial-and-error movement along x and y eventually locates the three holes (Figure 2, right), until the maximum shown in Figure 1 (right) is attained. The amount the mask must displaced to attain maximum brightness measures the x and y offsets of the signal from the reference.

Now assume one of the LEDs is off at either position $M_{1,1}$ or $M_{3,3}$. A maximum transmitted signal of 2 can be obtained at two different alignments: i) correct overlap or ii) a single element displacement in x and y as shown in Figure 2, left. Because the maximum achievable brightness is reduced from 3 to 2, we would know that one LED is off. Loss of one LED, however, introduces alignment ambiguity.
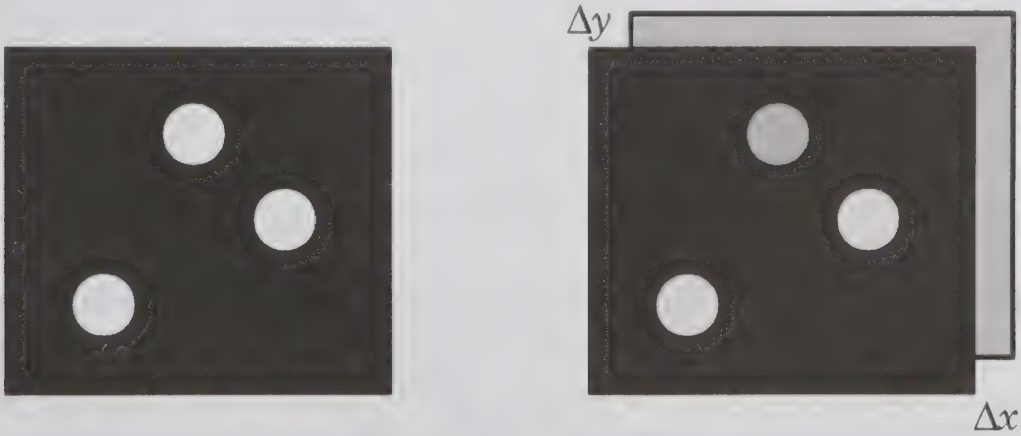

Figure 3: 3x3 Costas Array (left). Position offsets Δx and Δy between signal and reference frames are unambiguously located even with one LED off (right).

The Costas Array shown in Figure 3 (left) solves this problem. Holes and LEDs are located at array positions $M_{1,2}$, $M_{2,3}$, and $M_{3,1}$. If any one of the three LEDs is dark, the two remaining LEDs can be used to align the mask with no positioning ambiguity (Figure 3, right). Ambiguity returns if two or more LEDs

are off.

The 3x3 Costas Array assures alignment accuracy even when any one of the LEDs is compromised. Dark LEDs represent missing or corrupt data. This situation often occurs on noisy channels such as encountered in sonar, radar, and weak-signal communications. Maintaining critical time and frequency synchronization between transmitted and reflected signals in submarine sonar motivated research in this area by J.P. Costas [2].

In the above example, the Costas Array is used for two-axis (x, y) positional accuracy. In communication applications, the array dimensions are changed to time and frequency.

Costas Array in FT8. The square matrix representing the Costas Array is implemented with discrete time steps (columns) and frequency steps (rows). In FT8, there are 7 sequential time steps and 7 non-sequential frequency steps. The row values are the integers 3,1,4,0,6,5,2 and the time steps run from 1 to 7 as depicted in Figure 4. This is just one of the 200 possible 7x7 Costas Arrays [3]. The unique footprint of the Costas Array allows unambiguous alignment of the data stream at the receiving station. The decoder will look for the expected time sequence of symbols in the received message and attempt to establish the needed temporal and frequency synchronization between the two stations.
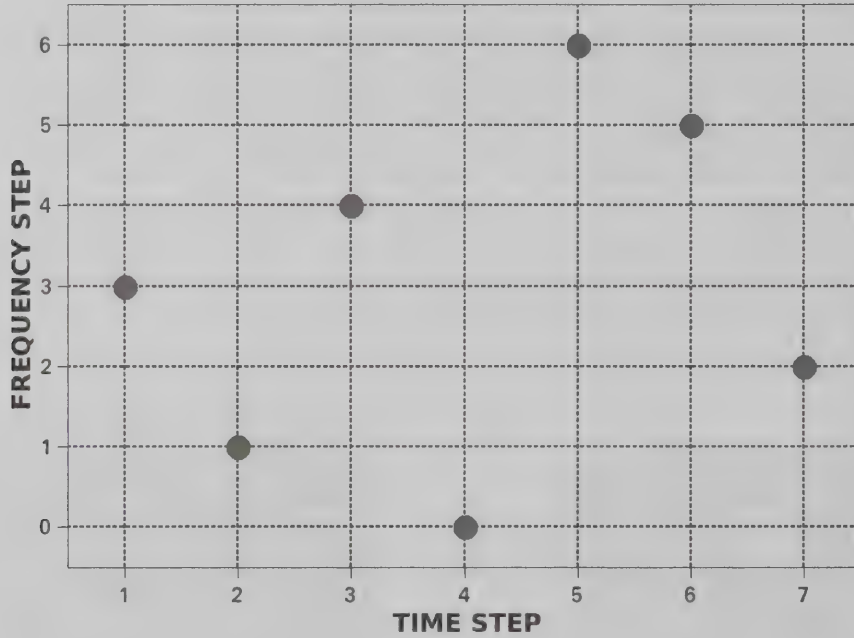

Figure 4: 7x7 Costas Array used in FT8 v2. The time and frequency steps are 160 ms and 6.25 Hz, respectively.

The 7x7 Costas Array is rendered by 7 of the 8 available FT8 tones, occupying 7 time steps. The sequence of tones is 3,1,4,0,6,5,2. Each tone is one symbol of duration 160 ms and 7 symbols require 1.12 seconds. To compensate for drift and fading that may occur over the 12.64 second duration of an FT8 transmission and to accommodate time synchronization offsets in the range: $-2 \leq \Delta t \leq +3$ seconds, the same Costas array is inserted at the beginning, middle, and end of each message. This means that 26.6% of each FT8 message is allocated to synchronization.

Figure 5 (left) is a time plot of the imaginary part of the 7x7 Costas Array sampled at 12000 S/sec. This signal modulates the audio carrier frequency f_c selected by the FT8 operator for transmission. In version 2.1 of WSJT-X, Frequency Shift Keying (FSK) has been replaced with Gaussian FSK (GFSK). Transitions
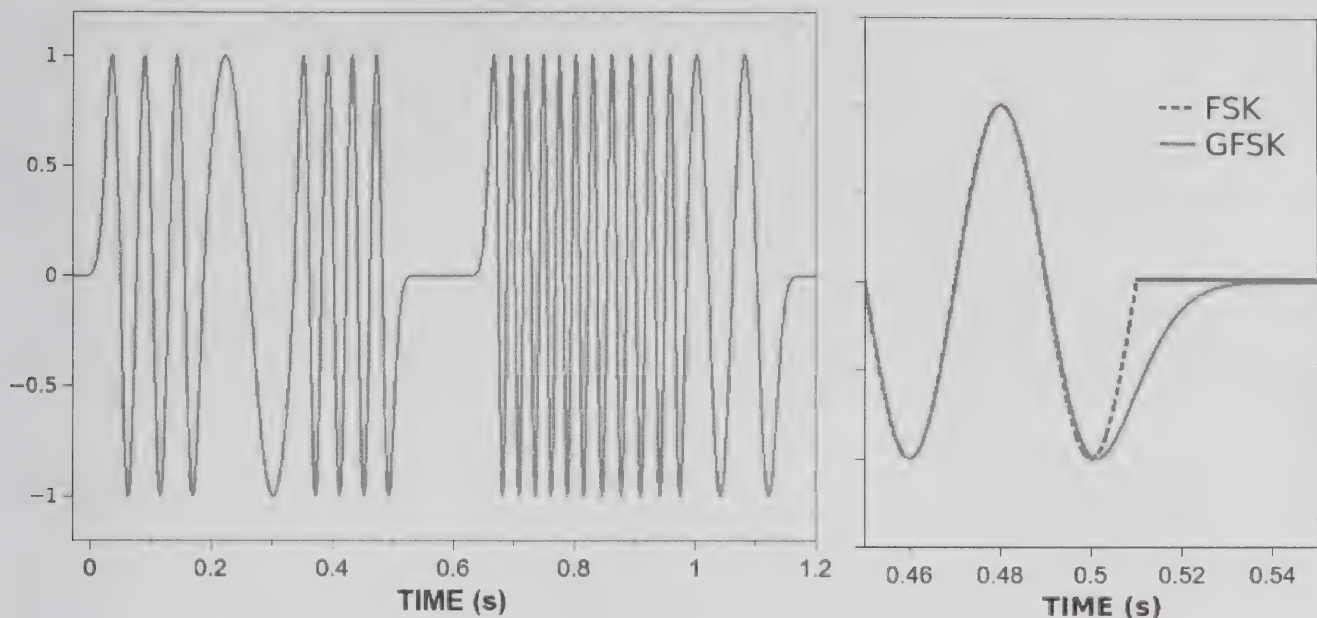
Figure 5: Left: The 7x7 Costas Array rendered as a sequence of GFSK symbols at 6.25 Hz multiples. The audio carrier frequency f_c has been set to zero for clarity. Right: Comparison of FSK and GFSK at the transition between Tone 4 and Tone 0.
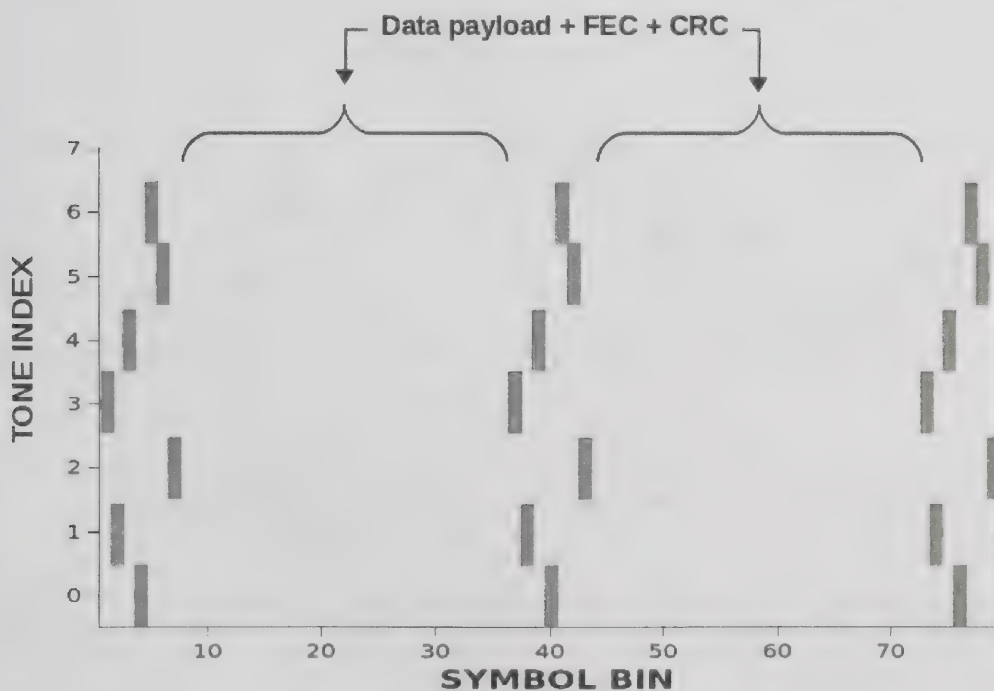

Figure 6: Arrangement of the 7x7 Costas Array in the transmitted FT8 signal. The synch tones occupy 21 of the 79 symbol bins. The information bits, Forward Error Correction, and Cyclical Redundancy Check are placed in the remaining 58 bins.

between the tones (symbols) are not as abrupt (Figure 5, right), resulting in a significant reduction of signal bandwidth [4].

GFSK modulation is identical for the 21 synchronization tones and 58 encoded message symbols. Legacy FT8 (in versions 1.9 and earlier) uses a 7x7 Costas Array that is the reverse sequence of the current version, ie. 2,5,6,0,4,1,3. This was changed in version 2 so that the new decoder could potentially recognize signals from earlier versions of the program. The location of synch tones, however, remains the same (Figure 6).

There is additional bandwidth required for each symbol (tone) to accommodate the baud rate, ie. 6.25 Hz. Tone 0 occupies frequency $f_0 = f_c \pm 3.125$ Hz. Tone 7 is $f_7 = f_c + 43.75 \pm 3.125$ Hz. The total FT8 signal resides in the frequency span: $f_0 = f_c - 3.125$ to $f_7 = f_c + 43.75 + 3.125$ Hz, which defines the 50 Hz modulation bandwidth.

Decoding. Signals may appear on a multitude of carrier frequencies anywhere in the receiver audio passband with an unknown time offset relative to the receiver clock. Figure 7 illustrates two trains of FT8 synchronization symbols. The audio carrier frequencies are located at Tone 0; the time offset between the two signals is slightly greater than one symbol bin.
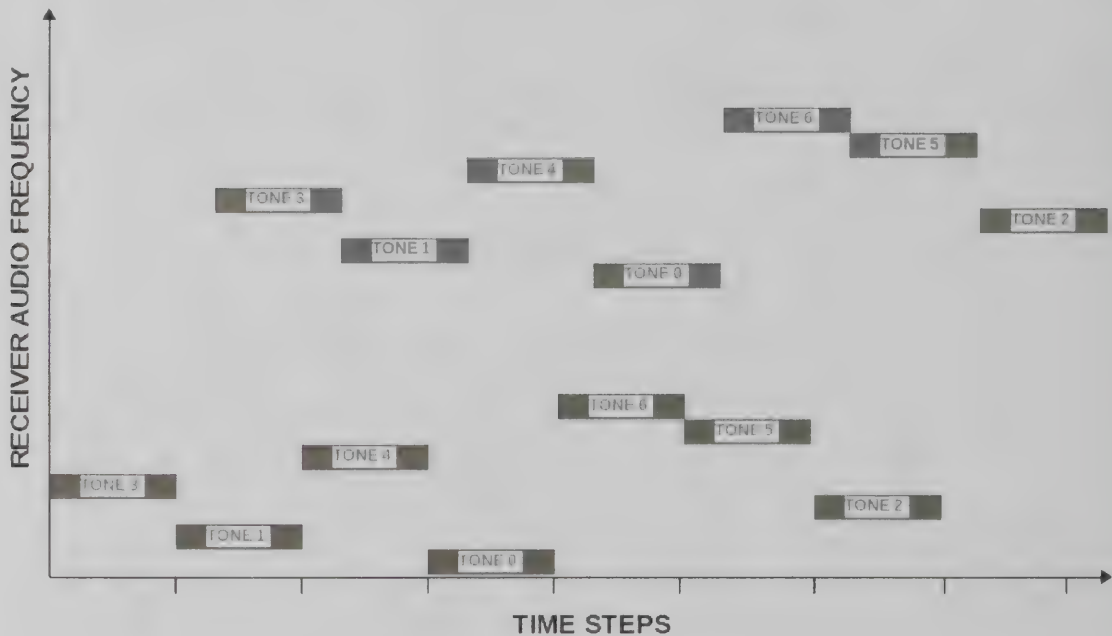

Figure 7: Costas Array symbol sequence for two FT8 signals in the receiver passband. The signals are offset in time by slightly greater than one symbol bin (0.16 s/symbol). Eight of the 79 symbol bins are shown.

The first task of the decoder is to establish time synchronization with any incoming data. This is accomplished in a series of operations depicted by the block diagram in Figure 8. Each block is described in more detail below.

The received FT8 signal is sampled at 12000 S/sec for 15 seconds, generating 180,000 16-bit audio samples. This corresponds to 180k x 16 = 2.88 Mb of data. Decoding does not commence until all the data has been acquired.

The decoder begins by searching the data for synchronization signals. This is handled by the *FORTRAN* subroutine sync8.f90 in the FT8 source code. The audio energy spectrum is calculated at sequential, partially overlapping time windows. The time increment is 1/4 of the duration of a single FT8 symbol,
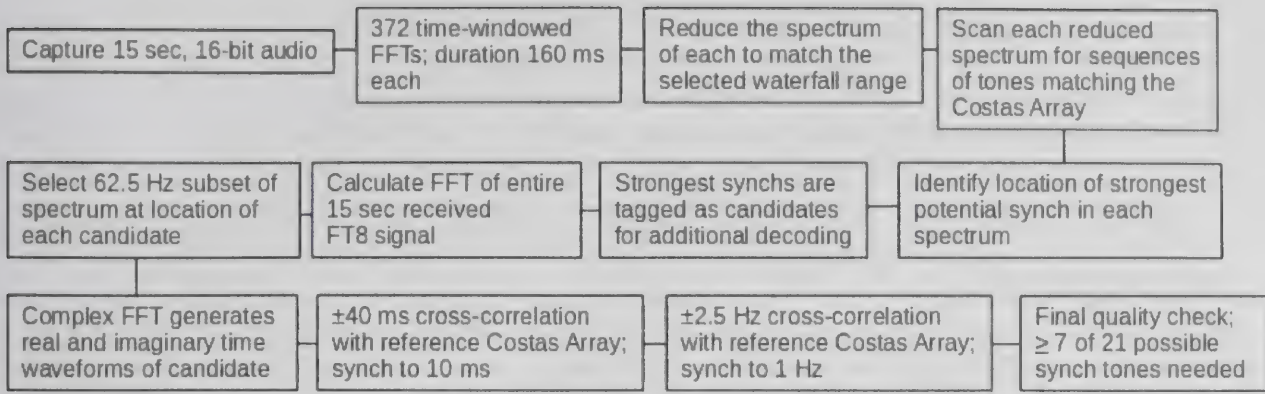
Figure 8: Block diagram of the FT8 synchronization process.

i.e. 40 ms. 372 individual spectra are obtained, with the last 120 ms of the 15 second data capture ignored. Spectra are generated by performing a sequence of 372 time-partitioned Fast Fourier Transforms (FFT_i where i=1,2,...372) with data collected from 160 ms time windows, but interleaved by 1/4 symbol. For example, FFT_1 is evaluated for the interval 0-160 ms, FFT_2 is 40-200 ms, FFT_3 is 80-240 ms, FFT_4 is 120–280 ms up to FFT_{372} for the final 160 ms time segment. This is illustrated for the first ~ 1 second of the received signal in Figure 9.

Each 160 ms sample interval is zero-padded to produce a 320 ms input signal for the FFT calculation; the FFT_i spectra span a frequency range from 3.125 Hz to 6 kHz. The operator sets the FT8 waterfall frequency range considerably smaller than this, eg. 200–2500 Hz, so only the calculated FFT range matching the waterfall is used.

The decoder next attempts to obtain time synchronization between the receiving station and a possible transmitting station at an audio baseline frequency f_c. There can be many received stations in the audio waterfall, each with a different f_c and time synch offset. The data at f_c is scanned for the correct synch tones employing 125 different start times t_0 in the range $t_0 = 0.5 \pm 2.5$ seconds, where the +0.5 second offset accounts for the delayed start of transmit. The time search increment is also 40 ms.

Starting at the lowest frequency that was set in the waterfall (eg. $f_c = 200$ Hz) the decoder sums the
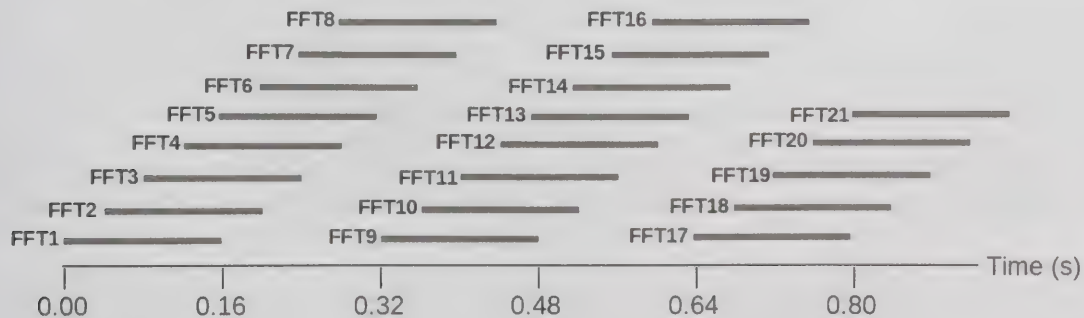
Figure 9: Searching for the synch signal at the start of the captured FT8 waveform. FFTs are calculated in the time intervals depicted by the horizontal bars. If the transmitter and receiver are perfectly aligned, the 7 elements of the first Costas array will be in the intervals FFT1, FFT5, FFT9, FFT13, FFT17, FFT21, and FFT25 (not shown). The decoder scans 125 different time offsets looking for the synch tones. The vertical axis in this plot is meaningless.

spectral energy density at the expected locations of the 7 Costas synch signals. As an example, when $t_0 = 0$ the energies at the following 7 frequency locations are summed: $f_c + 18.75$ Hz in FFT_1, $f_c + 6.25$ Hz in FFT_5, $f_c + 25$ Hz in FFT_9, f_c in FFT_{13}, $f_c + 37.5$ Hz in FFT_{17}, $f_c + 31.25$ Hz in FFT_{21}, and $f_c + 12.5$ Hz in FFT_{25}. These are slices of the time-partitioned energy spectrum corresponding to the Costas array sequence [3,1,4,0,6,5,2]. This synch sum t_a is normalized to the spectral content in all of the lowest 7 frequency bins for each of the 7 time intervals. The normalization sum t_{0a} is obtained by adding frequency bins 0–6 in FFT_1, FFT_5, FFT_9, FFT_{13}, FFT_{17}, FFT_{21}, and FFT_{25}. There are 7 components in the sum t_a and 49 components in t_{0a}.

This calculation is repeated at the expected location of the remaining two synch signals. For the present example with $t_0 = 0$, these are time positions 5.76 seconds (Costas tones expected in FFT_{144}, FFT_{148}, FFT_{152}, FFT_{156}, FFT_{160}, FFT_{164}, FFT_{168}) and 11.52 seconds (Costas tones expected in FFT_{288}, FFT_{232}, FFT_{236}, FFT_{240}, FFT_{244}, FFT_{248}, FFT_{252}). This produces synch sums t_b and t_c and normalization sums t_{0b} and t_{0c}, respectively. The decoder then generates sums $t = t_a + t_b + t_c$ and $t_N = (t_{0a} + t_{0b} + t_{0c} - t_a - t_b - t_c)/6$. t_N represents the averaged energy content of the 6 nominally empty frequency bins at each of the 7 time positions. The final normalized synch signal is $\text{S}_{abc} = t/t_N$.

For each f_c, the decode algorithm looks for the maximum total synch signal while scanning 125 time increments in the range $-2 \leq t_0 \leq 3$ seconds. For start times $t_0 < 0$, some or all of the Costas synch signal t_a will not be available. To accommodate this, a second normalized synch signal S_{bc} is calculated as above, except t_a and t_{0a} are ignored. The larger value of either S_{abc} or S_{bc} is recorded as the total synch signal at f_c for each of the 125 start times. Normalization allows S_{abc} and S_{bc} to be directly compared despite their different data counts.

The frequency f_c is incremented by 3.125 Hz and the above time synch calculation repeated. The frequency iteration process continues until the highest frequency that was set in the waterfall is reached. For waterfall audio in the range $f_c = 200$–2500 Hz, there will be 737 x 125 = 92,125 synchronization calculations performed for each 15-second FT8 receive interval.

A 2-dimensional array holds the best (strongest) total synch signal for each f_c and the corresponding index of its time offset. This array is sorted from weakest to strongest signals using the *FORTRAN* subroutine in indexx.f90 [5]. A baseline synch signal is established at the approximate midway point of the sorted array range; all synch signals in the array are normalized to this value. This means only about half of the captured audio spectrum will have synch signals with a normalized value ≥ 1.

Starting with the strongest normalized synch signal, the decoder checks for values ≥ 1.5. If the strongest normalized synch signal is < 1.5, the decoder exits. Signals above the 1.5 threshold are tagged as *candidates* for further decoding. The audio baseline frequency f_c, time synch offset, and normalized synch signal of each candidate are recorded. As many as 200 candidate signals can be acquired. If there are two candidates within 4 Hz of each other, however, the weaker candidate is discarded.

At this point, synchronization has been established to 3.125 Hz and 40 ms for every candidate signal in the waterfall display. Greater accuracy is possible using coherent detection with the *FORTRAN* subroutine ft8b.f90.

The first operation in ft8b is to reacquire the spectrum of the complete 15-second FT8 signal. This is accomplished with a call to the subroutine ft8_downsample.f90, which performs a FFT. A frequency resolution of 0.0625 Hz is obtained by zero-padding the time signal by 1 second to make 192,000 total time steps at 12000 samples/second.

There is a carrier frequency f_c associated with each candidate. The decoder locates f_c in the spectrum and slices out the portion $f_c - 9.375$ Hz to $f_c + 53.125$ Hz representing 1000 sample points. f_c is subtracted

from this array subset to baseline the spectrum at $f_c = 0$ Hz. An inverse FFT is then performed on the frequency-shifted spectrum to generate a complex time domain signal (i.e. in-phase and quadrature, phase-shifted $+90°$) without the carrier. The 16-second complex temporal waveforms have 3200 points, giving 5 ms resolution. Each candidate signal is re-cast in complex form to enable tighter synchronization and eventual decoding.

The decoder enters subroutine sync8d.f90 and generates a complex reference waveform with $k = 7$ symbols $R_{k,m}$ representing the FT8 Costas Array. Because the correlation calculations make a heavy demand on the processor, the sampling rate is reduced by a factor 60 compared to the waveform shown in Figure 5. The Costas Array used in this subroutine has $m = 32$ samples/symbol separated by 5 ms. GFSK smoothing is not effective at the coarser resolution, so the reference waveforms are rendered with standard FSK (upper 2 traces in Fig. 10).
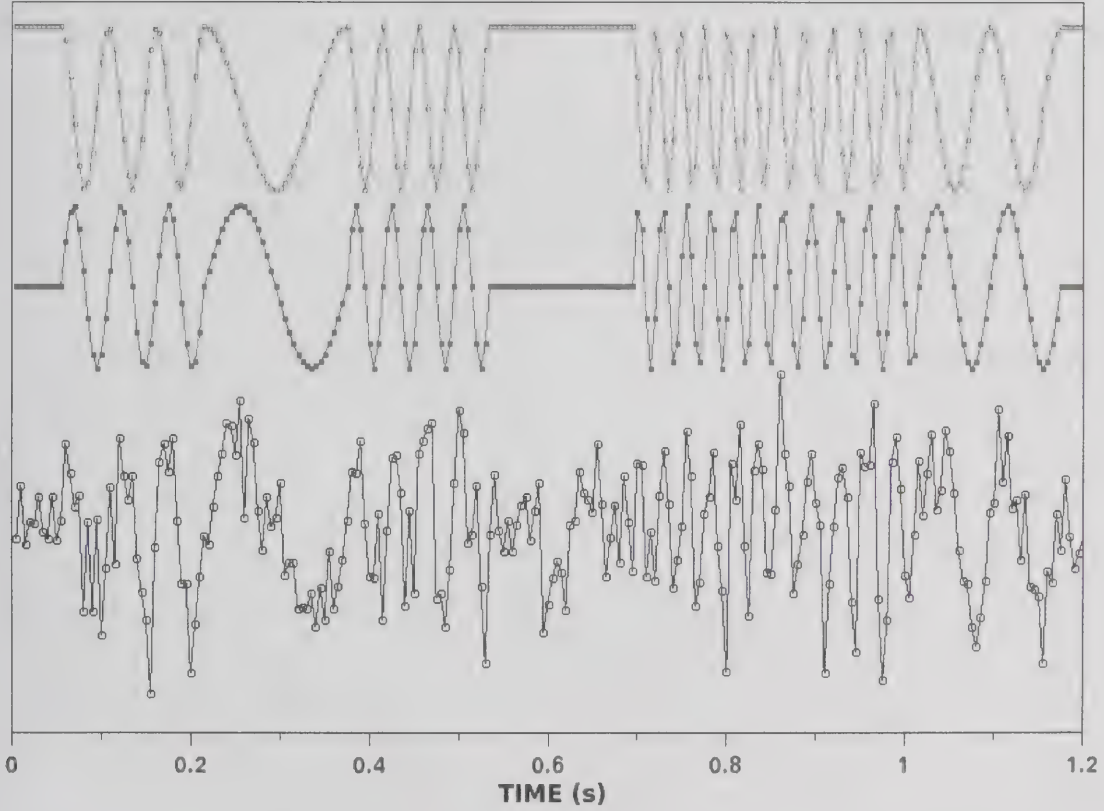

Figure 10: Real (top) and imaginary (middle) components of the reference complex Costas Array waveform with 32 points/symbol as generated by subroutine sync8d.f90. There is padding with Tone 0 on both sides for the cross-correlation calculations. The bottom waveform depicts the imaginary component of a simulated received signal with Gaussian amplitude white noise; signal-to-noise ratio is 2:1. Not shown is a similar noisy waveform for the real component. Waveforms are displaced vertically for clarity.

The candidate signal is trimmed to 3125 data points separated by 5 ms and contains up to three Costas Array synch waveforms $C_{j,k,m}$ located at the beginning (j=1), middle (j=2), and end (j=3). Each candidate has time synchronization already established to $\pm$ 40 ms, so it is only necessary to search within this 80 ms range with $n = 17$ time-step iterations of 5 ms/step. This is implemented with a cross-correlation between the received symbols C and reference symbols R in the Costas Array as follows. A complex product is calculated for each received symbol C at each Costas Array location j:

$$z_{j,k} = \sum_{m=1}^{32} C_{j,k,m} R^*_{k,m} \tag{1}$$

where each of the $m = 1-32$ elements (i.e. sample points) of a symbol C is multiplied with the corresponding element in the complex conjugate of the reference symbol R^*. The squared amplitude of each calculation in Equation (1) is summed to produce the aggregate synchronization signal:

$$T_n = \sum_{k=1}^{7} \left[|z_{1,k}|^2 + |z_{2,k}|^2 + |z_{3,k}|^2 \right] \tag{2}$$

T_n is evaluated at $n = 17$ different time alignments between C and R: $\Delta t = -40, -35, ...0... + 35, +40$ ms. At each time step n, $j \times k = 21$ individual correlations are performed, each with duration of one symbol, i.e. 160 ms. The decoder looks for the maximum value of T_n and sets this as the final time synchronization point.

Long duration correlations using the entire 7-symbol sequence of the Costas Array can, in principle, generate sharper peaks. This is not practical because signal coherence may be lost over the propagation path on a timescale approaching 1 second. Moreover, the carrier phase may be misaligned with respect to the reference Costas Array waveform anywhere in the range $0-2\pi$. The summed correlation power of individual symbols given by Equation (2) produces a useful peak without any knowledge of the carrier phase offset. Coherence is only required for the duration of each 160 ms symbol.

The operation described above can be illustrated by calculating cross-correlations of the reference Costas Array with simulated received signals. Representative waveforms are shown in Figure 10. The top and middle waveforms are the real and imaginary (quadrature) components, respectively, of the reference Costas Array, i.e. without noise. The bottom waveform shows the imaginary component with additive Gaussian white noise at a signal-to-noise ratio of 2:1 as might be encountered with a weak received signal [6].

Calculated cross-correlations for a single FT8 Costas Array [3,1,4,0,6,5,2] are shown in Figure 11, representing just one term on the right side of Equation (2). A noise-free sequence of 7 symbols establishes the synchronization point to better than 10 ms (solid curve). The three dashed curves show the effect of additive Gaussian white noise at a signal-to-noise ratio of 2:1. The correct synchronization point is still located with an accuracy of $\pm$ 10 ms. This illustration is for a single Costas Array sequence; there may be as many as two more additional signals z_j available on the received signal to produce a useful maximum in T.

Fine frequency adjustment is performed with a cross-correlation using the same subroutine sync8.f90 and procedure outlined above. The frequency is scanned in $n = 11$ increments of $\Delta f = 0.5$ Hz/step over the range $\pm$ 2.5 Hz. This corresponds to an incremental phase scan of $\Delta\phi = -4.5°, -3.6°, ...0°... + 4.5°$ in an attempt to peak the cross-correlation T_n. This second cross-correlation brings frequency synchronization to within 1 Hz. The time and frequency corrections determined by the two cross-correlations are then applied to the corresponding candidate data set to establish the desired synchronization.

Before attempting to decode the message contained in the 58 information bins, the ft8b subroutine makes a final quality check of the optimized synchronization. A sequence of 79 time-partitioned FFTs is performed at the time slot of every tone in the entire candidate signal. Each 160 ms symbol has 32 sample points that are used by the FFT algorithm to calculate its complex spectrum. The first 8 points in the FFT data array correspond to the energy content in the 8 FT8 tones; only these 8 frequency bins are of interest. The FT8 tone with the largest magnitude is assigned to that bin. In this way, the 21 synchronization bins are assigned one of the FT8 tones. The information bins are evaluated with a separate procedure.
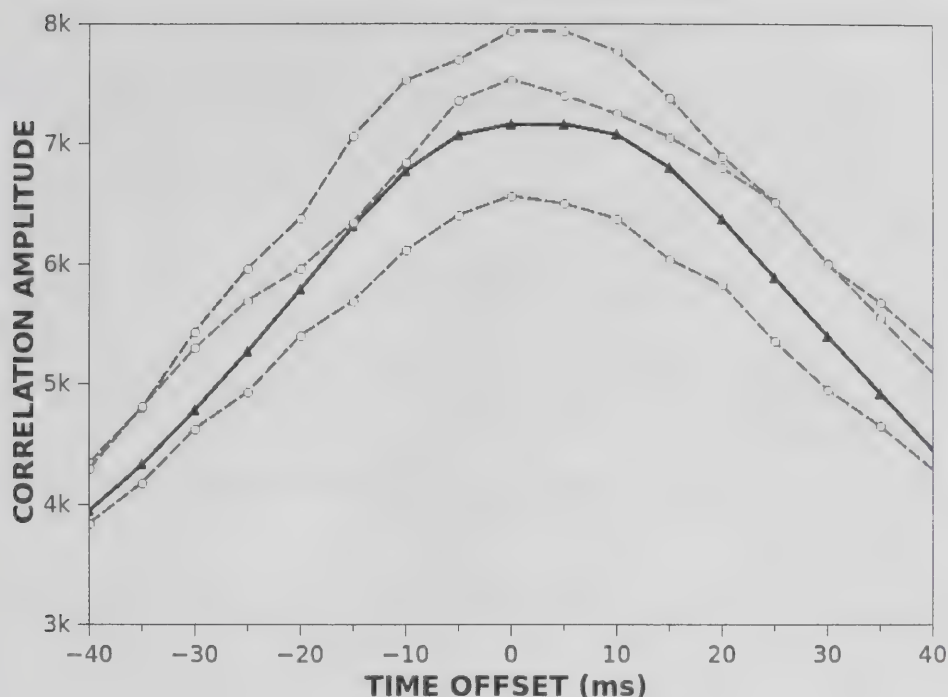
Figure 11: Simulated cross-correlations using a single FT8 Costas Array [3,1,4,0,6,5,2]. A noise-free, perfectly phase-aligned correlation is depicted with the solid black line. The dashed curves are for signals having carrier phase randomly aligned with the reference plus additive Gaussian white noise; signal-to-noise 2:1.

The next step is to compare the tones that have been assigned to bins 1–7, 37–43, and 73–79 with the tones expected for the Costas Array, i.e. the tone sequence 3,1,4,0,6,5,2 as shown in Figure 6. Three error-free sequences results in the maximum possible 21 matches. Because of the powerful Forward Error Correction present in the coded messages, the decoder can reliably proceed with as little as 7 matches. Fewer than 7 matches will cause the decoder to give up and evaluate the next candidate.

Acknowledgements. Helpful dialogue was provided by J. Frazier (KC5RUO), P. Karn (KA9Q), S. Franke (K9AN), and J. Taylor (K1JT).

Disclaimer: This is not official documentation for the WSJT-X open source project. It represents the author's best understanding of the FT8 synchronization scheme. There may be errors and misconceptions.

Updates to this document will be posted at: www.sportscliche.com/wb2fko/tech.html.

References

[1] J. Taylor, S. Franke, B. Somerville "Work the World with WSJT-X, Part 2: Codes, Modes, and Cooperative Software Development", QST (Nov. 2017)

[2] J.P. Costas, "A study of a class of detection waveforms having nearly ideal range-doppler ambiguity properties", Proc. IEEE **72**, 996 (1984).

[3] S.W. Golomb and H. Taylor "Construction and properties of Costas arrays", Proc. IEEE **72**, 1143 (1984).

[4] J. Taylor, S. Franke, B. Somerville "The FT4 Protocol for Digital Contesting", WSJT-X Developer Technical Note (April 22, 2019).

[5] B.P. Flannery, S. Teukolsky, W.T. Vetterling, W.H. Press, *Numerical Recipes in FORTRAN: The Art of Scientific Computing*, Cambridge:NY (1997).

[6] The simulation ignores leakage resulting from windowing in the inverse FFT.

WSPR in an educational Project

How to simply program a WSPR transmission with an Arduino?

Anthony LE CREN, F4GOH
f4goh@orange.fr

Abstract

Since several years I have been trying to design projects to promote HAM radio, and especially for young people who want to pass their radio license. But most of the time, projects are complicated and require a good level of electronics. The article described below is for people who know to blink a led with an Arduino and wiring a breadboard.
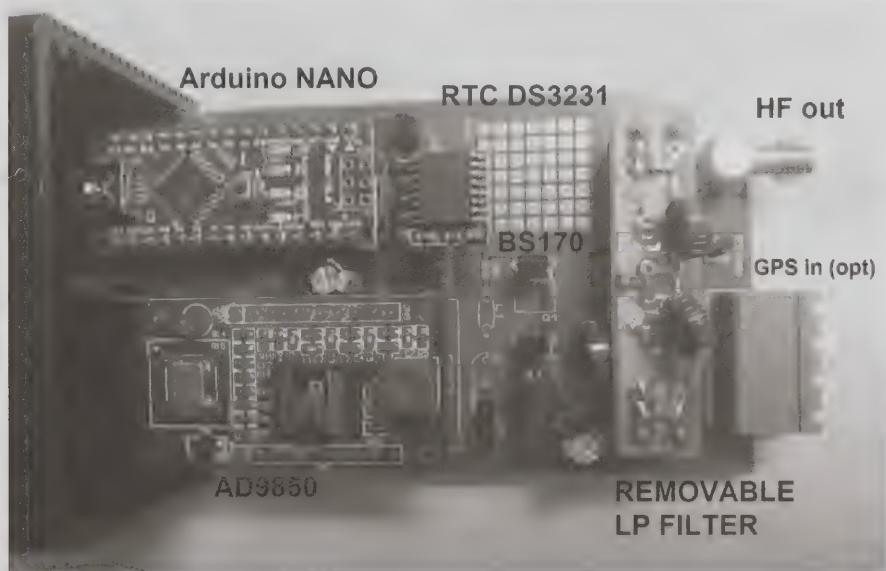
Introduction

The idea is to generate WSPR signal with an AD9850 controlled by an Arduino, A DS3231 RTC synchronize time to transmit WSPR. The level of programming has been limited to the easiest, in fact, a simple loop makes it possible to transmit the WSPR signal. In addition the assembly can be wired on a breadboard step by step. You will learn how to program a DDS, make filters and setting the transmission frequency.

WSPR beacon features: Arduino nano (atmega328P) based microcontroller :
- Support Arduino IDE 1.0+ (OSX/Win/Linux)
- Power via USB or External Source + 5v
- 2 I/O Pins (for GPS receiver)
- Removable filter
- AD9850 DDS
- RTC DS3231
- BS170 power amplifier 0.1W
- Fit in Aluminium Instrument Box Enclosure Case 100x66x43
- PCB available

1 Description

RTC DS3231

Mini USB 5V

Arduino Nano

DDS ad9850

PA BS170

D
G
S

Low pass filter

antenna
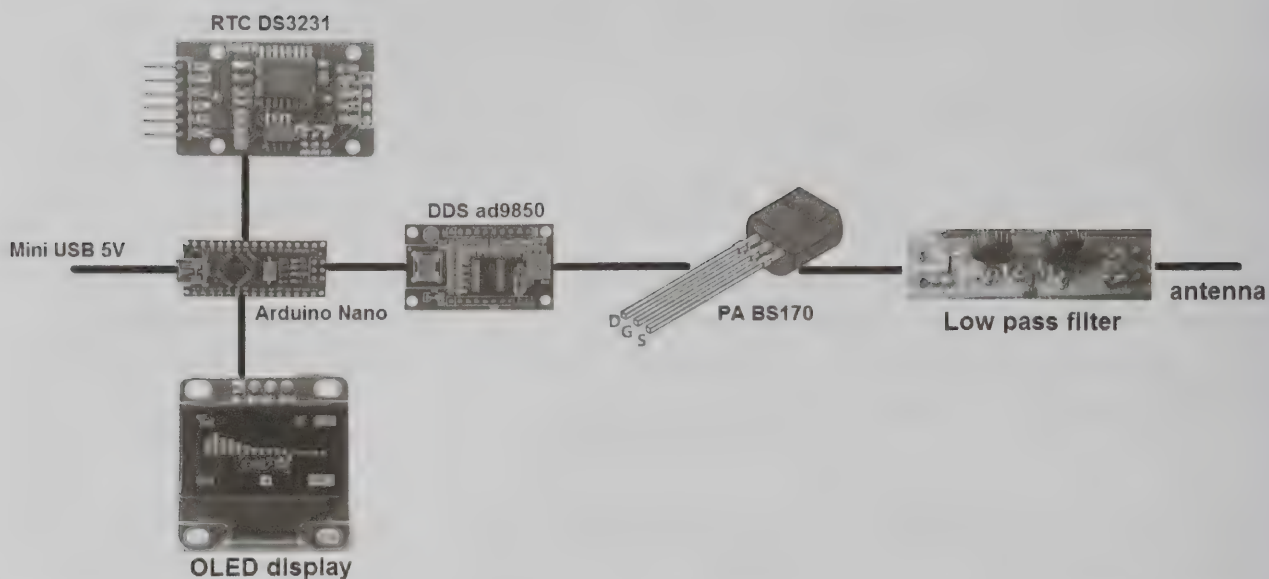
OLED display
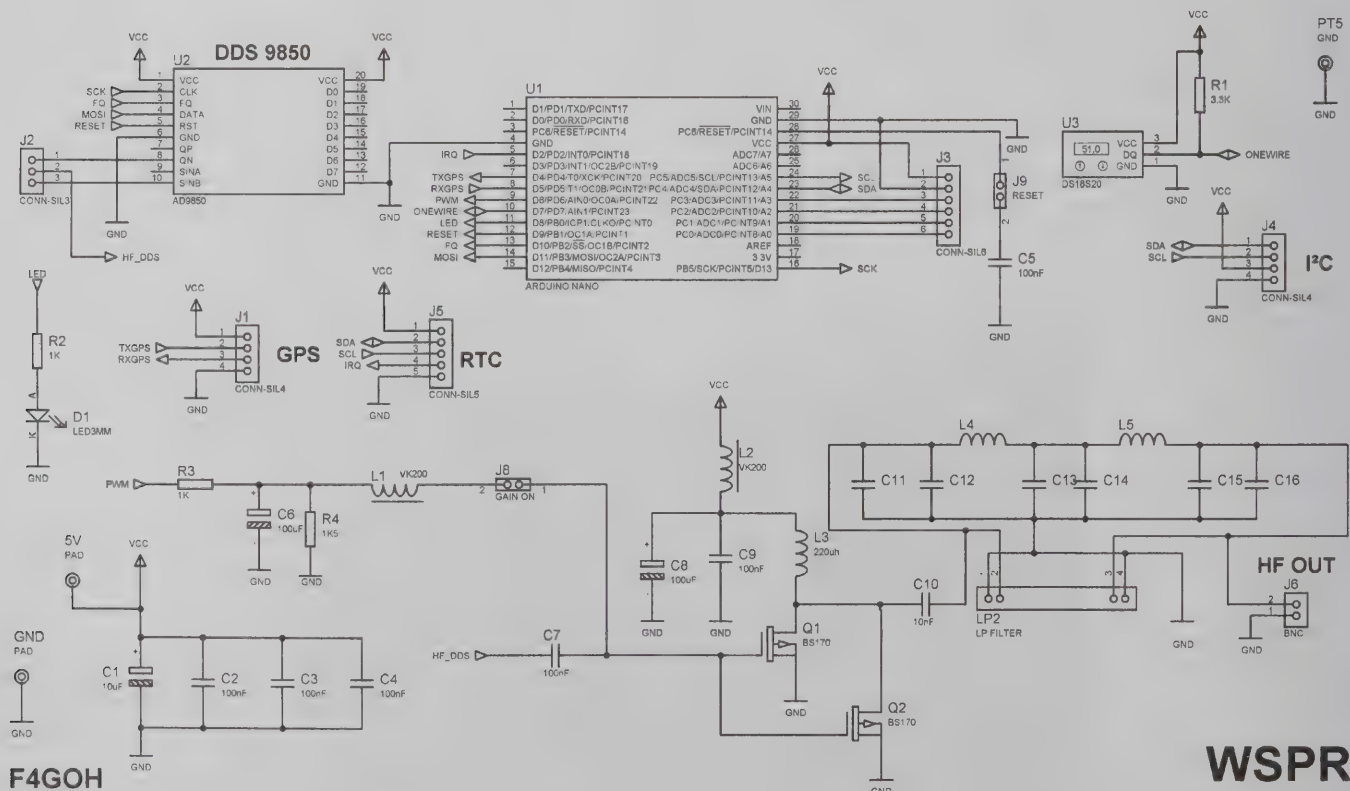
The real-time clock synchronizes Arduino to transmit every even minute. The Oled screen is optional and simply displays the time. DDS (Direct Digital Synthesis) generates an RF signal. The transistor BS170 is used as a class E amplifier and requires a low-pass filter.

Schematics:

DDS 9850

GPS

RTC

F4GOH

I²C

HF OUT

WSPR

Two ways to use bs170 :

– use QN (J2) DDS output and replace C7 capacitor by a strap. don't put R3,C6,R4,L1, and left j8 open (this is the configuration i used). Adjust DDS 9850 trim to have a ~ « square wave » at the QN output. When transmission is OFF QN voltage should be 0V. Use any scope, you can find GND ·connection at PT5.

– use SINB(J2) and apply voltage polarization on BS170 gate. (put R3,C6,R4,L1 and strap J8) then apply analogWrite command on nano pin 6. (to adjust gain by software). if gain value is too high bs170 may be destroyed.

You can find a pad space to experiment your own stuff and change bs170 to another transistor.

J1 connector could be dedicated for GPS input. But it can be used for another things like external lowpass filter commutation.

Ds1820 is planned just for fun, to send temperature by RTTY or PSK modulation.
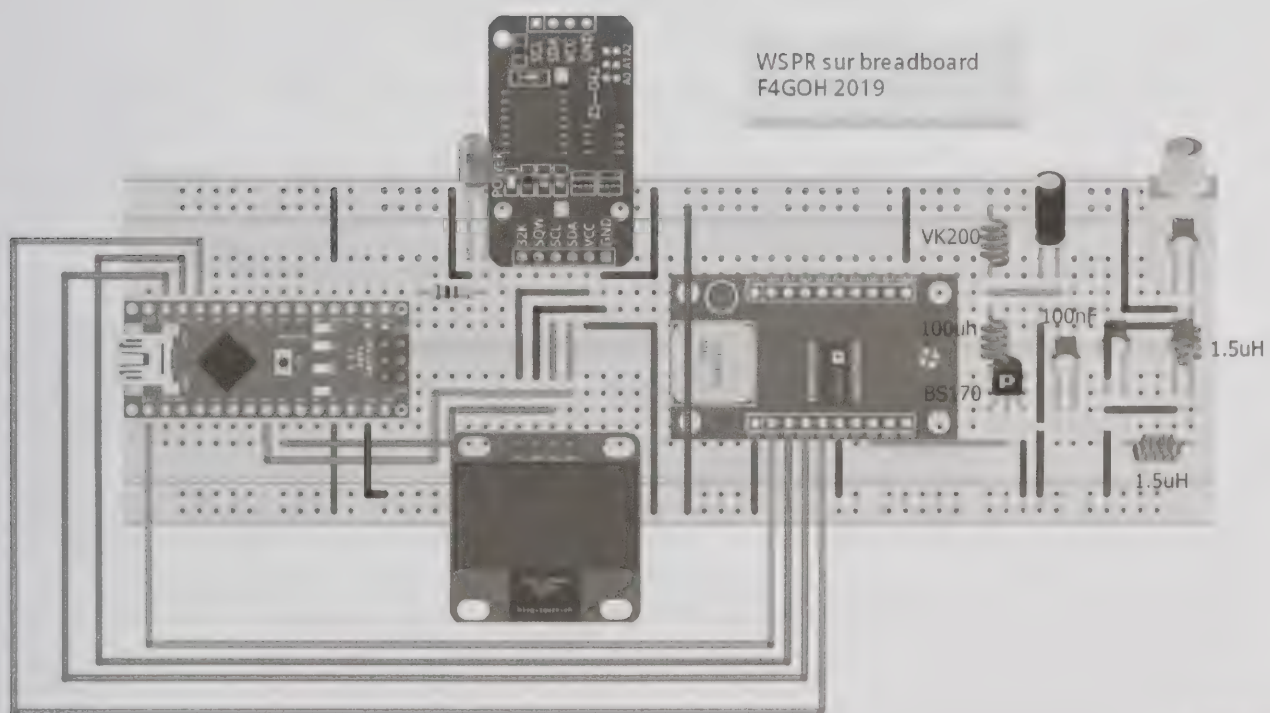
The transmitter can be wired on a breadboard

2 How to program AD9850 DDS?
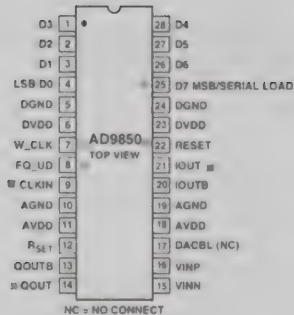

ANALOG DEVICES

CMOS, 125 MHz Complete DDS Synthesizer

AD9850

FEATURES
- 125 MHz Clock Rate
- On-Chip High Performance DAC and High Speed
- Comparator
- DAC SFDR > 50 dB @ 40 MHz A$_{OUT}$
- 32-Bit Frequency Tuning Word
- Simplified Control Interface: Parallel Byte or Serial Loading Format
- Phase Modulation Capability
- 3.3 V or 5 V Single-Supply Operation
- Low Power: 380 mW @ 125 MHz (5 V)
             155 mW @ 110 MHz (3.3 V)
- Power-Down Function
- Ultrasmall 28-Lead SSOP Packaging

FUNCTIONAL BLOCK DIAGRAM
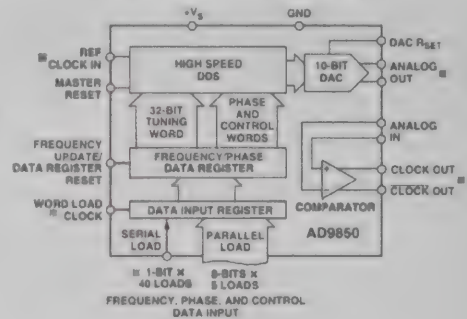
Arduino nano uses its SPI (Serial Peripheral interface) to update the frequency in the DDS

Table III. 8-Bit Parallel Load Data/Control Word Functional Assignment

Word	Data[7]	Data[6]	Data[5]	Data[4]	Data[3]	Data[2]	Data[1]	Data[0]
W0	Phase-b4 MSB	Phase-b3	Phase-b2	Phase-b1	Phase-b0 (LSB)	Power-Down	Control	Control
W1	Freq-b31 MSB	Freq-b30	Freq-b29	Freq-b28	Freq-b27	Freq-b26	Freq-b25	Freq-b24
W2	Freq-b23	Freq-b22	Freq-b21	Freq-b20	Freq-b19	Freq-b18	Freq-b17	Freq-b16
W3	Freq-b15	Freq-b14	Freq-b13	Freq-b12	Freq-b11	Freq-b10	Freq-b9	Freq-b8
W4	Freq-b7	Freq-b6	Freq-b5	Freq-b4	Freq-b3	Freq-b2	Freq-b1	Freq-b0 (LSB)

The frequency is not update in HZ. You must send a 32-bit word proportional to the frequency as shown in the formula below.

$$\underbrace{f_{OUT}}_{\text{output Frequency}} = (\underbrace{\Delta \, Phase}_{\text{32 bit word}} \times \underbrace{CLKIN}_{\text{ref Frequency } 125mhz})/2^{32}$$

$$\Delta \, Phase = \frac{2^{32} \times f_{OUT}}{CLKIN}$$
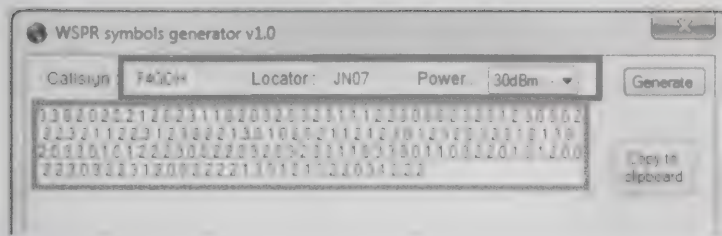
```
  setfreq(double f, uint16_t p) {
            deltaphase;

  deltaphase = f * 4294967296.0 / (125000000 + factor);
  for (int i = 0; i < 4; i++, deltaphase >>= 8) {
    SPI.transfer(deltaphase & 0xFF);
  }
  SPI.transfer((p << 3) & 0xFF) ;
  pulse(FQ_UD);
}
```
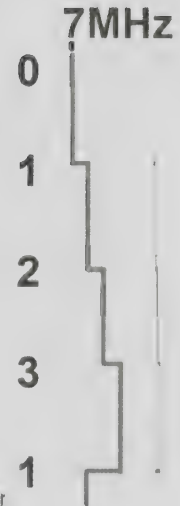
3 WSPR encoding

7MHz

```
0

1

2

3

1
```

WSPR symbols generator v1.0

Callsign: F4GOH   Locator: JN07   Power: 30dBm   [Generate]

[symbol grid image]

[Copy to clipboard]

base : 7.000.000 Hz
0 : 7.000.000 Hz
1 : 7.000.001.46 Hz
2 : 7.000.002.92 Hz
3 : 7.000.004.38 Hz

Resulting in 162 sequential symbols each with a value from 0 to 3

Modulation

Each symbol represents a frequency shift of 12000 / 8192, or approximately 1.46Hz, per Symbol Value giving four-level Multi-FSK modulation. The transmitted symbol length is the reciprocal of the tone spacing, or approximately 0.683 seconds, so the complete message of 162 symbols takes around 110.6 seconds to send and occupies a bandwidth of approximately 6Hz.

To generate symbols, execute WSPR symbols generator :

- Put your callsign
- Put your locator
- And finish by power (20 dBm for 0.1W)
- Click Generate
- Paste symbols on wsprSimple.ino program.

wsprSimple | Arduino 1.8.9

Fichier Edition Croquis Outils Aide

wsprSimple

```
#define RESET 9  // or 10
#define frequence 7040100 //base freq


long factor = -1500;    //adjust frequency to wspr band
int secPrec = 0;


int wsprSymb[] = {3, 3, 0, 0, 0, 2, 0, 2, 1, 2, 0, 2, 3,
                  2, 2, 3, 0, 1, 3, 2, 0, 3, 3, 2, 3, 2,
                  0, 1, 3, 2, 1, 2, 1, 2, 2, 2, 3, 0, 0,
                  2, 1, 0, 1, 2, 0, 3, 3, 2, 2, 0, 2, 2,
                  };
```
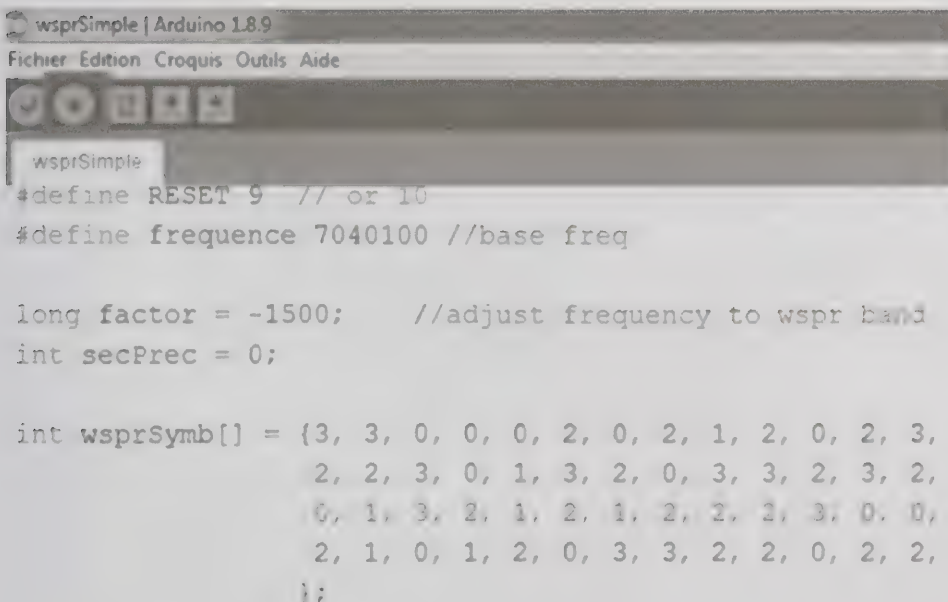
The FOR loop simply transmit the WSPR signal :

```
    sendWspr(long freqWspr) {

   a = 0;
 for (int element = 0; element < 162; element++) {     // For each element in the message
   a = int(wsprSymb[element]); //   get the numerical ASCII Code
   setfreq((double) freqWspr + (double) a * 1.4548, 0);
   delay(682);
   Serial.print(a);
   digitalWrite(LED, digitalRead(LED) ^ 1);
 }
 setfreq(0, 0);
 Serial.println("EOT");
}
```

4 Build a low pass filter (40 meters, 7.2 Mhz)

The best way is use svcfilter designer. This software calculates inductors and capacitors for any bandwith.

Analysis page
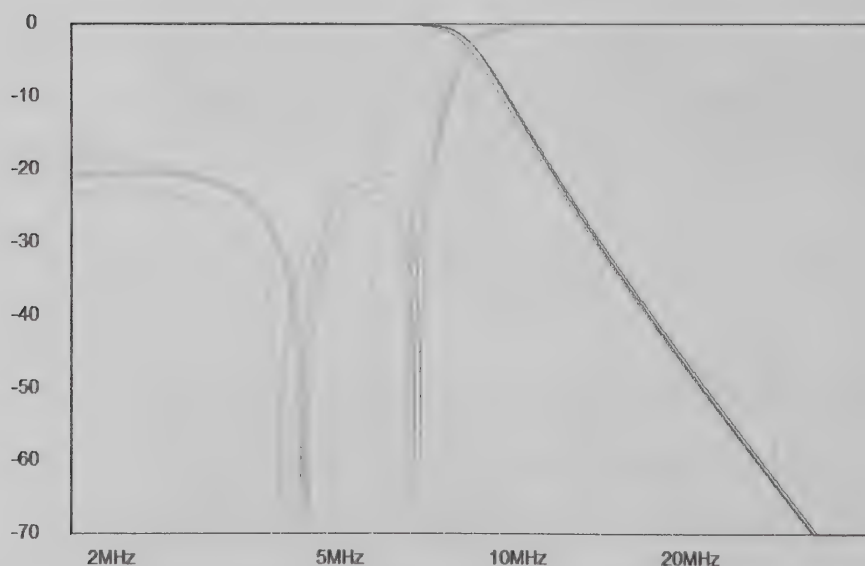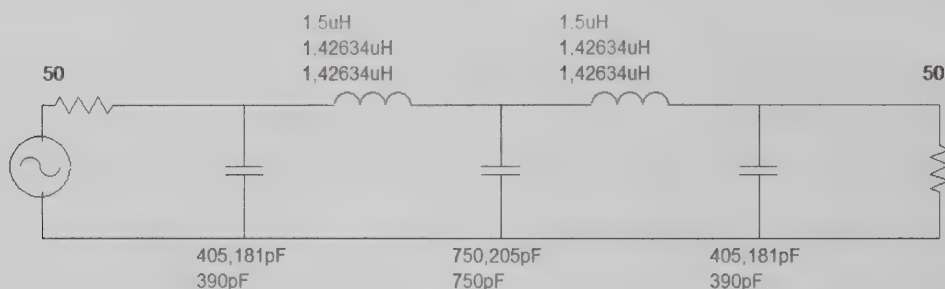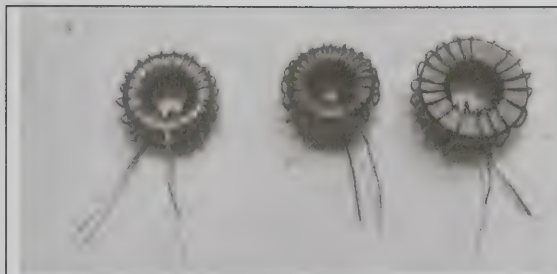
Use mini tore calculator software to check numbers of turns:

t30-2 :19 turns
ft37-43 (BS170 drain inductor): 18 turns
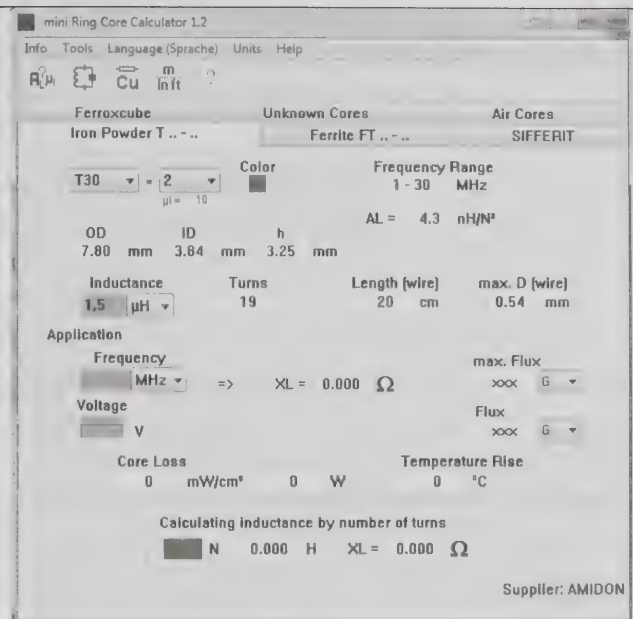
5 Conclusion

It was a pleasure to study DDS and WSPR modulation. This project was designed essentially for makers. Adapt it as you want. There are more screenshots on my web page. Reports are incredible with an 5V USB power.

References
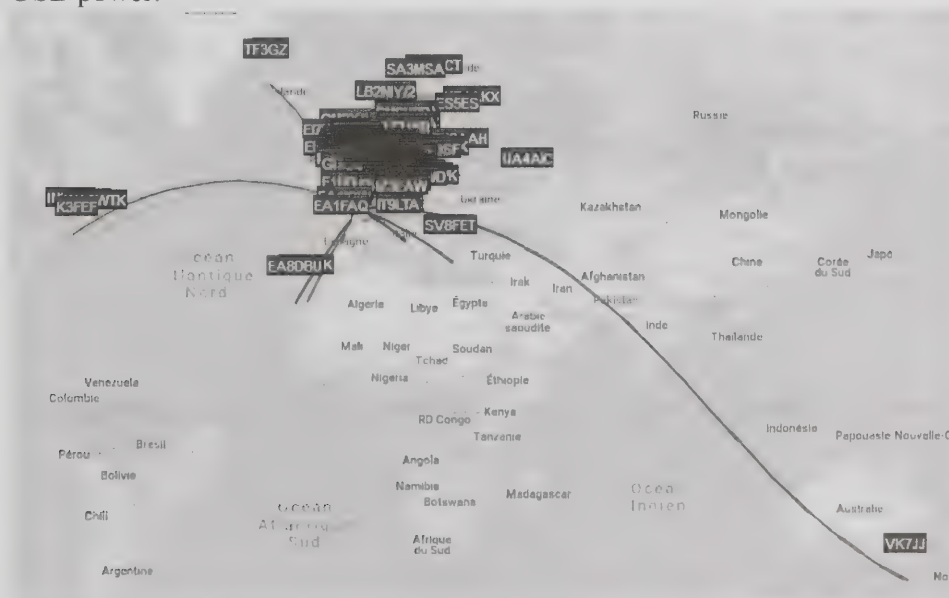
https://hamprojects.wordpress.com/
https://github.com/f4goh/WSPR
http://www.g4jnt.com/wspr_coding_process.pdf
http://tonnesoftware.com/svcdownload.html
https://constructions.f6fkn.com/downloads/minirk12-install.exe
https://kitsandparts.com/ (toroids)

Portable Audio Frequency-Shift Keying Sensors using a Hamshield mini

Nolan Pearce*, KE8JCT, Stephen S. Hamilton*,†, KJ5HY and Kate J Duncan*,†, KB2ZOO

United States Military Academy*, Army Cyber Institute†

{ nolan.pearce, stephen.hamilton, katherine.duncan}@westpoint.edu

Abstract

The newly developed Hamshield mini and commercially available electronic devices integrated with the Hamshield can be used to create an auto-reporting ham radio motion detector. Open-source Arduino code, a Passive Infrared (PIR) sensor, and looped audio frequency shift keying (AFSK) transmissions were assembled to create a low-power, low-cost, open library, small form-factor device that expands upon current automatic remote beacon sensors. In this paper, we detail an open-source amateur radio QRP VHF/UHF packet radio using the Hamshield. Furthermore, the compatibility with Arduino single-board microcontrollers will enable current amateur radio technologies.

Key Words

QRP, Arduino, Packet, AFSK, Automation

Introduction/Background

The Hamshield mini is a new crowd-funded piece of amateur radio technology that appears constrained only by its user's imagination; the mini is capable of data, voice, CW, remote control, and telemetry all within a form factor of approximately a USB stick. This new device is part of a larger trend in technology where fast, cheap, open source projects replace legacy hardware systems as is often performed with the Raspberry Pi. This "rapid prototyping" utilizes readily available feedback to provide quick solutions to simple problems and for which no conventional solution exists. This project is intended to provide substance and evidence for ad-hoc solutions to unusual and emergent situations, while utilizing the Hamshield.

Solution

The Hamshield mini is integrated with a PIR sensors to form a mobile movement detector. This device automatically relays a predetermined AFSK message on the 2m ham band when triggered by a change in the PIR sensors output voltage. This message can then be decoded on a remote ground station far from the sensor itself. The PIR sensor's low voltage operation and small form factor made it easily compatible with the end goal of the project. To create this code, the Hamshield's AFSK example code on github [1] was downloaded and modified for this project. This was then combined with another open-sourced demonstration of the PIR sensor [4] and modified to run in a continuous loop. Figure 1 describes the schematic of the system.
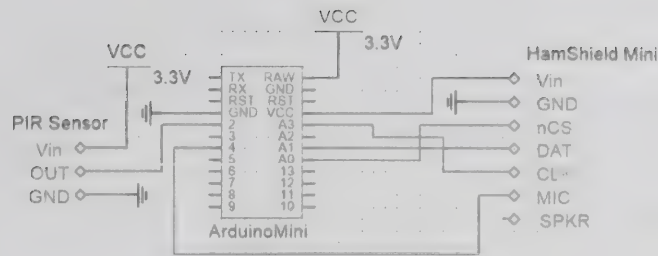
Figure 1

Design

Several competing factors created the final design of the device. The Hamshield's low-cost and aspects such as low power consumption, small form factor, and open-source code, were factors in deciding on it for the use as our disposable remote sensor. First, the goal of the project was to develop a low-cost, open source solution for an AFSK mobile messaging movement detector. Arduino pro minis have common libraries and low operating voltage (3.3 Volts) and cost as little as $10. Next, the system needed to be battery-powered with an extended lifespan (24 hours). Anker phone batteries provide long battery life and are the same size of a mini breadboard. Additionally, the Hamshield operates at a power output of 200 milliwatts [1], which is desirable for line-of-sight QRP (low power) communications. Finally, the device must have as small form factor; although the Hamshield mini was prototyped with longer wires plugging directly above the Arduino on the breadboard this form factor can be reduced.
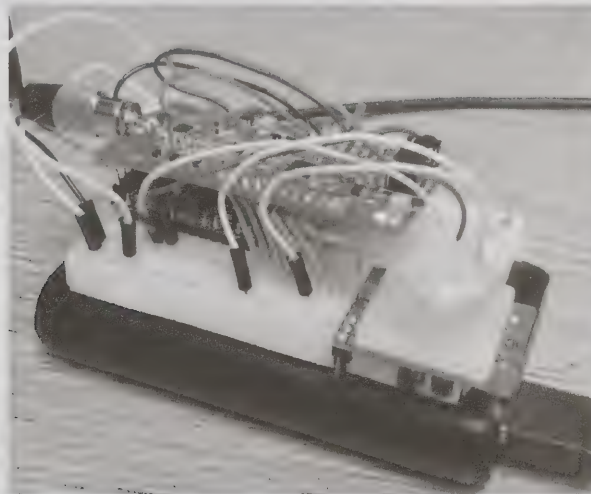
Figure 2

Figure 2 shows the prototyped product; clearly visible are the battery, Arduino mini, PIR sensor, and Hamshield over top of the microcontroller. The antenna, an ANT-500, acted as the largest component of this system: a promising feature of this project is the ability in integrate more features onto the Arduino to increase capabilities, such as a distance measurer or other telemetry devices.

Results

The finished project operates effectively with low power consumption, a small form factor, and easily obtainable components (the total system cost was under $70). The code remains open-source and compatible with Arduino technology [3]. Due to the operating voltage of the PIR sensor (using low-power 3.33 Volts instead of 5 Volts), its sensitivity results in minimal false positives for movement. However, even with the low radiated power of the system, and the low-profile ANT-500 antenna, the testing was successful. Measured tests showed the furthest transmission range to be approximately 50 meters, which could be increased by using a higher gain antenna. The decoding station operated from a portable laptop with QTMM (freeware AFSK decoding software) and an RTL-SDR. Figure 3 shows the output from the decoder.

Input: Microphone (High Definition Aud

```
11:08:01$ fm *-15 to *-0 UI^ PID=F0
         *
11:08:08$ fm KG7OGM-15 to *KG7O-0 UI^ PID=F0
         Hi there.*
11:08:22$ fm KG7OGM-15 to *KG7O-0 UI^ PID=F0
         Hi there.* KG7OGM,KG7OGM,:Hi there.*
11:09:28$ fm KG7OGM-15 to *KG7O-0 UI^ PID=F0

11:09:42$ fm KG7OGM-15 to *KG7O-0 UI^ PID=F0
```

Decoder running
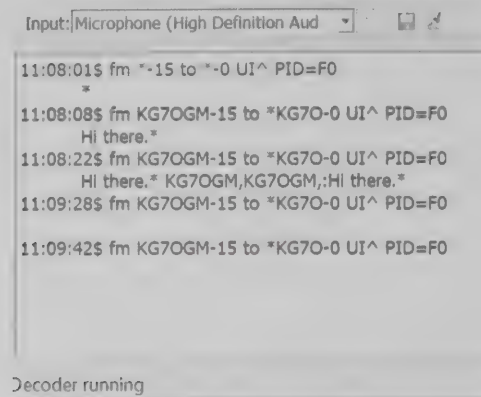
Figure 3

The small size of the device enables the system to be integrated into various situations, and with proper construction and enclosure, the device can be used in most environments. The "fish-eye" lens of the PIR sensor also allows for a wide coverage area (5m), with just a single device.

Conclusion

The increase in crowdfunded and open source radio technology represents a change in the mindset of engineer solutions for communications; oftentimes these rapid prototyping devices offer quicker feedback than conventional research and development methods. This project stands as a demonstration towards the usefulness of this technology. Using commercially available components, the Hamshield mini performs effectively as a remote PIR sensor. In further studies, the Hamshield's capabilities can be expanded. The Hamshield, could operate as an APRS telemetry station, due to the AX.25 packet protocols and its compatibility with Arduino, with customize transmission data packets. This Hamshield based APRS telemetry station also would be more cost effective compared to consumer-grade equipment. Multiple nodes and digipeaters can expand the propagation of the warning signals and increase their effectiveness. Overall, these commercially available component-based systems that operate on open-source code can be expanded upon and researched for their novelty in solving amateur radio problems.

References

[1] Halverson, Casey and Morgan Redfield. "Hamshield Mini – Enhanced Radio Devices." *Enhanced Radio Devices Website,* Enhanced Radio Devices, 2018. http://bit.ly/2IA0ysU

[2] Halverson, Casey and Morgan Redfield. "Hamshield AFSK Serial Messenger." *Enhanced Radio Devices Github,* Enhanced Radio Devices, 2019. http://bit.ly/3390nya

[3] Pearce, Nolan. "Motion Detector." *KE8JCT Github,* 2019. https://github.com/KE8JCT/MotionDetector

[4] "Using a PIR w/ Arduino." *Adafruit Explore and Learn*, Adafruit Systems, 28 Jan 2014. http://bit.ly/2K7ggxD

An FPGA Learning Experience: SPI Interface to Max10 FPGA

Gregory Raven, KF5N

greg.electricity@gmail.com

July 31, 2019

Abstract

A story of learning FPGA technology with the evolution of a simple project.

1 Introduction

This is the "Golden Age" of SBCs (Single Board Computers) and microcontroller boards. Walk into your favorite brick-and-mortar bookstore (if you can find one), and check out the magazine shelf. You will find several magazines, or "book-a-zines" dedicated to "Arduino" or "Raspberry Pi". On the bookshelves you will find a few books on Arduino and RPi, maybe even one for BeagleBone.

Or look in the magazine "Make", or use their "Makers' Guide to Boards":

`https://makezine.com/comparison/boards/`

Using the above webpage, you can select a "Type" of board from three categories:

- Microcontroller

- Single Board Computer

- FPGA

While there are dozens of boards listed, there are only three shown for "FPGA". This list is not entirely accurate, as more FPGA boards do exist, however, it gives an idea of the relative scarcity of FPGA boards.

There are a ton of resources online, and dozens of books and magazines on SBCs which you can find at the local bookstore. The ARRL store sells several SBC themed books as well, but the web store shows none on FPGA.

I think it is fair to say that FPGA technology has not made it very far into the ranks of hobbyists. Amateur radio experimenters have certainly been pioneers in FPGA, with numerous "Software Defined Radio" projects going back to the era when the devices were really expensive. FPGA is an amazing technology, a sort of "3D printer of digital electronics". Perhaps there are other applications of FPGA within the amateur radio world in addition to SDR which can be explored?

I wanted to experiment with this FPGA stuff! The current crop of devices and boards has lowered the cost of entry. This is my story of a first project in FPGA. Hopefully it can be shown that FPGA projects are within reach, and other hams can be encouraged to try working with this fascinating technology.

2   Why?

Why work with FPGAs? Aren't SBCs good enough?

I like to think of FPGAs as a sort of "3D printer for electronics". A loose analogy yes, but the point is that FPGAs allow you to create new digital circuits at will. It is HARDWARE not software! This gives you the power to create multiple specifically targeted digital machines which can work independently of one another. This is the capability that SDR uses to crunch DSP math very efficiently while handling data flow simultaneously. A list of FPGA applications from Wikipedia:

https://en.wikipedia.org/wiki/Field-programmable_gate_array#Common_applications

FPGAs will almost certainly be used in combination with conventional "hard" computing devices (like your laptop or Raspberry Pi). Think of an FPGA as a capability you can add to your SBC to expand its capabilities into high-efficiency computing and the real-time domain.

3   Where to Start

When I began my FPGA quest, I was already familiar with the most common SBCs and microcontroller boards, the Arduino, the BeagleBone, and the Raspberry Pi. It a good thing to have some experience developing with an SBC before attempting to tackle FPGA. In my case, I spent quite a bit of time developing several projects on the BeagleBone Black. Many of the skills learned working with SBCs will apply to FPGAs.

One of the ways I went up the learning curve on SBCs was this lecture series by Bruce Land of Cornell University:

https://www.youtube.com/watch?v=FYv6JLOvpgo&list=PLKcjQ_UFkrd4z2qstandljtVhC3u+sCTpk

This course uses a PIC32 microcontroller board. However, the material in the lectures is generic enough to apply to any board which can be programmed in C. Even better, a second course covers FPGA!

https://www.youtube.com/playlist?list=PLKcjQ_UFkrd7UcOVMm39A6VdMbWWq-e_c

and the matching website:

http://people.ece.cornell.edu/land/courses/ece5760/

Watching a few of the youtube videos gave me a pretty good idea of what was involved in FPGA work. I didn't use the same development board as used in this course, but I did use another Intel FPGA based board. So the development tools and general flow of working with the FPGA are similar.

You will need to use a variety of resources to answer questions and solve problems as you go up the FPGA learning curve. It seemed to me that most of the effort required to learn FPGA is in handling a large

quantity of details. You will need to spend a lot of hours viewing videos, reading, and taking notes. I think that FPGAs require at least a little more effort than working with SBCs.

4 Your FPGA Manufacturer Web Site

You will need to get an account at your FPGA manufacturer's website. I used an Intel based board, and the account was free. On the Intel site, you will have access to large amounts of learning resources for FPGA. Be prepared to spend many hours watching videos and studying documentation, whatever FPGA you have chosen. Intel has good material, and you can learn a lot! A recommended starting point is the video series "Become an FPGA Designer in 4 Hours". The 4 hours part is perhaps a bit optimistic, but it will give you a good early acceleration:

`https://www.intel.com/content/www/us/en/programmable/support/training/course/odswbecome.html`

5 Github Repository for this Project

The documentation and code for this project is located in this git repository:

`https://github.com/Greg-R/spiavalonfpga`

6 Choosing a Development Board

The first FPGA development board I purchased was the Numato Lab "Elbert V2" ($29.95) This board has the Xilinx XC3S50A Spartan 3A FPGA device containing 1584 logic cells and 54 KB RAM. The board has a nice selection of peripherals which can be driven by the FPGA:

- 16 MB Flash Memory
- USB 2.0 interface for Flash programming
- 8 LEDs
- 6 push buttons
- DIP switch
- VGA connector
- Audio connector
- SD card adapter
- 3x 7 segment LED
- 39 IOs

Note that the USB capability of this board is for Flash programming only. It is not a general purpose interface to the FPGA.

The reason for choosing the Elbert V2 was the book "Programming FPGAs" by Simon Monk. The book features the Elbert, along with the "Mojo" and "Papilio" boards. This book is one of the very few "hobbyist" style books I could find.

If you have a specific application, then it will be straightforward to decide if the development board meets your requirements. For a person motivated to learn FPGA without without a specific application in mind should choose a board with more peripheral devices than less. Also note that the example above has 39 IOs, which allows for additional peripherals to be installed. If the IO pins are in the format used by Arduino, the addition of peripherals will be easy to accomplish.

I was able to install the development software for the Elbert and work through a few of the projects in the book. However, I found the development tools lacking in one area. I was interested in using a more modern version of Verilog, called "SystemVerilog". The development tools for the Elbert V2 do not support that, so more searching was required.

My motivation for SystemVerilog was to try some of the new features, including those used for "testbenches" (simulation). That is only an expression of my particular interest, as the Verilog used in the Xilinx tool for this device is fine and can be used for maximum benefit with this device. This does, however, indicate that the development environment which mates up with the FPGA device is as important as the device itself. So before you choose a board, be sure to download and install the development tools first. Look at the documentation and decide if you will be comfortable with that particular development tool capabilities.

Looking around a bit more, I found this board after viewing Bruce Land's great lecture series on youtube:

https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=165&No=836

The DE1-SOC is used in Cornell's "ECE 5760 Advanced Microcontroller Design and system-on-chip" course.

The board requires the "Quartus" development tool, which met my requirement for SystemVerilog. However, the FPGA on this board is well beyond a beginner! This board uses an advanced "System On Chip" (SOC) which is a combination of microcontroller (dual-core ARM Cortex A9) and FPGA in a single device. Price is $249, which seemed a little steep for a project which might not work out. The added complexity of the ARM processors, having to deal with an unfamiliar distribution of Linux running on the ARM processors, and the all of the extra work involved seemed like too much. Indeed this board is immensely capable, so maybe I will come back to it in the future!

Fortunately the same company, Terasic, makes boards with less complex FPGAs (and less expensive). Searching their site, I found this board, the DE10-Lite ($85):

https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=234&No=1021

This looked like a good choice, and I purchased it. Later when doing internet searches, I found there are plenty of example projects already available for this board. It seems to be at moderately popular in the academic world, and some courses have been based on this board. The MAX10 FPGA is well supported, although it is not included in the latest version of the Quartus development tool (latest version supported 2018).

A good feature of this board is the Arduino compatible expansion header. In general, the DE10-Lite has been easy to work with, and seems to be robust. I've been using it for many months and it is still alive!

7   A Project Inspiration: JTAG to FTDI SPI Interface

The inspiration for this project came from a a search at github.com on DE10-Lite:

`https://github.com/hildebrandmw/de10lite-hdl`

This is a nice collection of work done with the DE10-Lite board for academic purposes. What really got my attention is the project play_gif:

`https://github.com/hildebrandmw/de10lite-hdl/tree/master/projects/play_gif`

The interesting feature of this project is the usage of the USB to load data (animated GIF image file) to the FPGA. So this is a data pipeline from a Linux desktop to the FPGA which is built into the board! This met my requirement that I be able to control the FPGA remotely from a desktop (or SBC) computer.

The interface is via "JTAG", which is typically used as a debugging interface. It is not specifically intended for mass data transfer, but in this case it was pressed into service.

The interface is a bit clunky to use. It requires a "TCL Server", and a running instance of the Quartus development tool! Not exactly what I was looking for, but I got the demo to work easily! It is very nicely done work demonstrating several features of FPGA technology.

The way it works is conceptually simple. The FPGA part of the project implements a VGA[1] interface to the connector on the DE10-Lite board. The image loaded from the desktop computer is sliced into its constituent "frames". Another interesting aspect of the project is the interface to the SDRAM of the DE10-Lite which is a 64MB external part on the board. The sliced-up image is loaded into the SDRAM, and then another control module pages the VGA output through the memory. Thus you see the animated GIF displayed on the monitor. Really you are seeing in a very direct manner the data loaded into the SDRAM. Cool!

7.1   IP and Platform Designer

First, a little bit of FPGA jargon. "Intellectual Property" (IP) in the context of semiconductor devices is a block of circuitry which has been heavily engineered and refined to perform some particular function. It could be patented or otherwise protected from duplication by competitors. Due to the way integrated circuits are manufactured, blocks of "IP" can be added to the silicon and be expected to perform to the IP owner's specifications. Typically IP can be included as part of a design kit, or it can be paid for with a license fee.

IP is good because it can reduce engineering design effort, improve performance, and enhance quality. The trade-off is license fee cost, and you don't necessarily get exactly what you want.

---

[1]VGA is a relatively simple video standard which seems to be common on many FPGA development boards.

In our case, we are given a whole bunch of IP for free that we can experiment with! This is bundled into "Platform Designer" which is a tool-within-a-tool in the Quartus design suite.

To do justice to this there should be an entire section on "Platform Designer". I will summarize here. There are excellent video Platform Designer tutorials which you can access if you register for a free account at the Intel web site.

"Platform Designer" is a building-block system. You get a library of IP, along with a mechanism to hook them together. The design is bundled into a "Qsys" file. Let's have a look at the Qsys part of the play_gif project:

Figure 1: Play_gif project Qsys file

The upper left corner of the GUI is the library of IP. Some of the categories:

- Basic Functions

- DSP

- Memory Interfaces

- Processors and Peripherals (including a "Nios" processor)

- Memory Interfaces and Controllers

There is a sort of "sub-library" called "University Program" which includes:

- Audio and Video

- Bridges

- Clocks

- Communication

- Generic IO

- Memory

The project used this IP:

1. ALTPLL Intel FPGA IP

2. JTAG to Avalon Master Bridge

3. DRAM Controller Intel FPGA IP

4. External Bus to Avalon Bridge

5. Clock Bridge

6. (Parallel IO) Intel FPGA IP

The above IP can be seen in the column "Name". There are two instantiations of the Parallel IO. In the column "Connections" can be seen the graphical interconnections between the IP blocks. Connections are made by simply clicking on the circles at the intersections of the "wires" between the IP. Thus an entire system can be assembled using this GUI. No writing of Verilog required! The project does include some hand-written Verilog. This "Qsys" design is "dropped in" to the project as a Verilog module.

Here is what the system looks like:
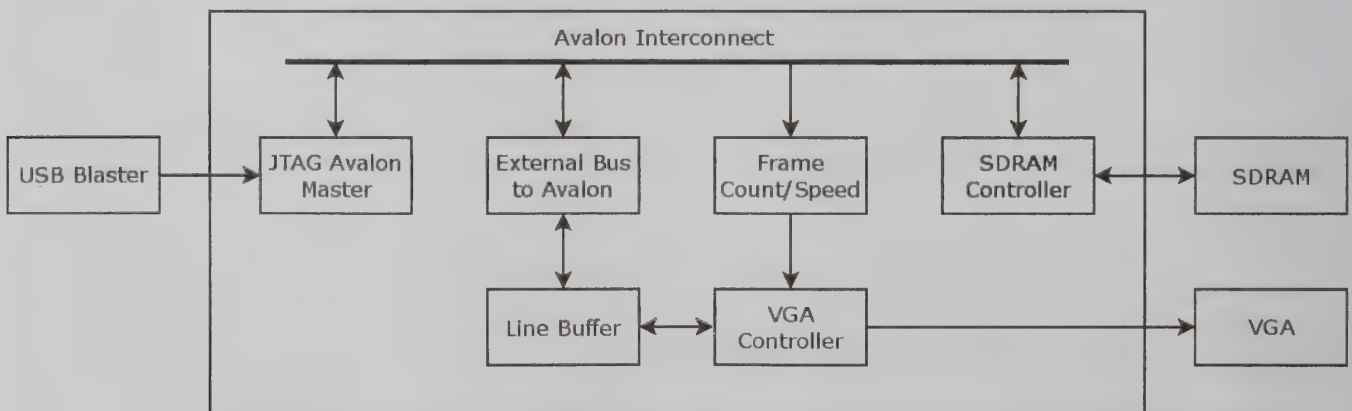
Figure 2: Play_gif JTAG Interface System Diagram

What this diagram does not show is the requirement for a running Quartus and a TCL/JTAG server program.

The "Avalon Interconnect" deserves some explanation. This is a system bus used in the MAX10 FPGA. From the Avalon Interface specification:

> Avalon® interfaces simplify system design by allowing you to easily connect components in Intel® FPGA. The Avalon interface family defines interfaces appropriate for streaming high-speed data, reading and writing registers and memory, and controlling off-chip devices. Components available in Platform Designer incorporate these standard interfaces. Additionally, you can incorporate Avalon interfaces in custom components, enhancing the interoperability of designs.

It is an internal bus standard used to connect Avalon bus masters and slaves. So it is a single-click process to connect Avalon components in Platform Designer. It is really amazing what you get for such little effort!

This project is interesting, and shows a path to communication between a desktop computer and the FPGA. However, the JTAG + Quartus + TCL/JTAG Server is cumbersome. A SPI to Avalon bus IP component is listed in the catalog. What if the JTAG and development tools could be replaced by something simpler like a SPI bus?

So that is what evolved into my "introductory FPGA project". The revised system diagram:
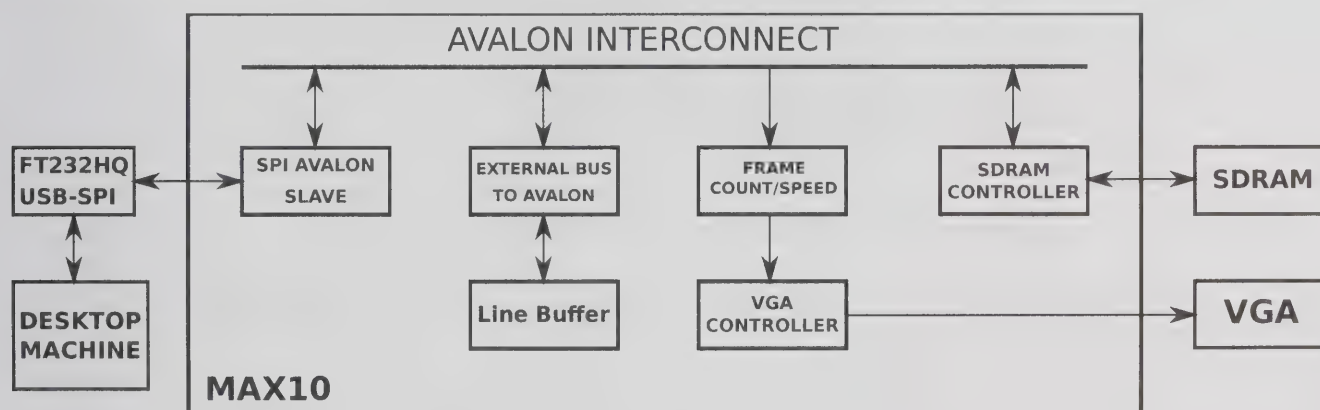
Figure 3: Play_gif with SPI System Diagram

The significant change on the FPGA is the swapping of the JTAG Avalon bus master with the SPI-Avalon slave. This was not entirely a drop-in replacement, as there were changes to reset and clock connections in addition to swapping JTAG to SPI components. But it is easy, and the swapping can be done in a couple of minutes.

External to the DE10-Lite board, there is a USB to SPI adapter board. This board is based on the FT232H chip by FTDI. This can be bought from eBay for about $10. Search for "ft232 spi" and you will find several options. I recommend one with headers to allow it to be plugged into a common breadboard. The FTDI device requires a shared library (libMPSSE) to be installed:

`https://www.ftdichip.com/Support/SoftwareExamples/MPSSE/LibMPSSE-SPI.htm`

Other changes required for SPI:

- The ports on the QSYS module changed (JTAG -> SPI), thus the Verilog module in which it is instantiated required minor changes. This was done with the text editor feature of Quartus.

- Another change is required to the FPGA pins. The new SPI bus must be routed to some easily accessible header on the DE10-Lite board. Since the board has an Arduino compatible header, and this header has a standard set of four pins for SPI, those pins were used. The details can be seen in the DE10-Lite manual provided by Terasic. The "Pin Planner" tool was used to make the changes.

- The "Synopsys Design Constraints" (.sdc) file was updated to incorporate the SPI bus.

Here is the rapid-prototype breadboard hook-up:
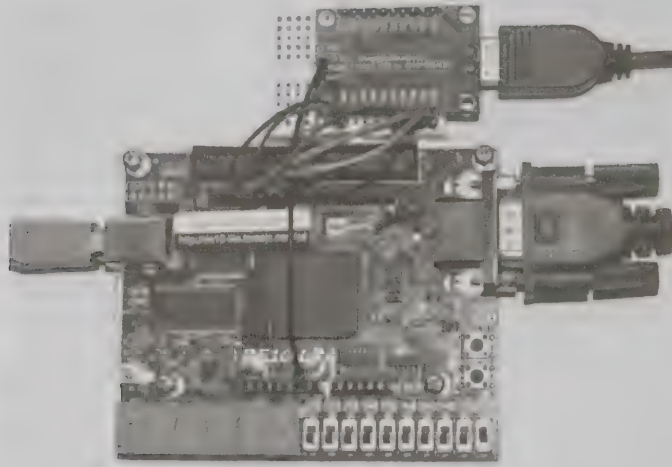
Figure 4: DE10-Lite Connected to FTDI SPI Breakout

In spite of the length of the breadboard jumper wires, the SPI interface performed remarkably well right up to the FTDI clock limit of 30 MHz.

8   SPI Driver in Julia Programming Language

The github project from which mine was derived uses a programming language called "Julia" in the JTAG server and for image data processing:

https://julialang.org/

This was a language I had heard about, but I had never tried it. To duplicate and run the original project, I had to install the language binary. The program does the processing of the GIF image, and then sends the data to the FPGA via the JTAG server and USB-to-JTAG interface.

It was simple to install the server and run the Julia program. It all worked first time! Later, I installed the Atom IDE which has a Julia plug-in. It's great!

Rather than reinventing the wheel, I decided to use the image processing portion of the Julia program. However, how to drive the FTDI SPI device? I needed something to replace the JTAG server.

The FTDI USB-to-SPI device is supported with a C shared library. This library has the initialization, read, write, and shutdown command necessary to work with the device.

Fortuitously, the Julia language includes the capability to call C library functions in a very direct way! I was skeptical at first, but I quickly had the SPI device's initialization function running and returning with no error. The other required functions were quickly added. I now had full control of the SPI bus from the command line!

When I say "command line", in the case of Julia I am referring to the "Read Eval Print Loop", called the REPL. This functionality is similar to Python, and is my favorite way to develop code. I also used the "Atom" IDE, which has a plug-in for the Julia language. It is a new language, and has a few quirks like all of them do, but so far I am impressed!

At first, Julia was also used to access a shared library which was taken from an Altera demo project of the SPI-Avalon bus master. This library was responsible for reading and writing "Avalon Packets". This is the protocol used by the Avalon bus. I was able to successfully translate the Altera library to Julia. This is working well and the system is able to read and write to the SPI and thus the FPGA Avalon bus, the parallel ports, and the SDRAM.
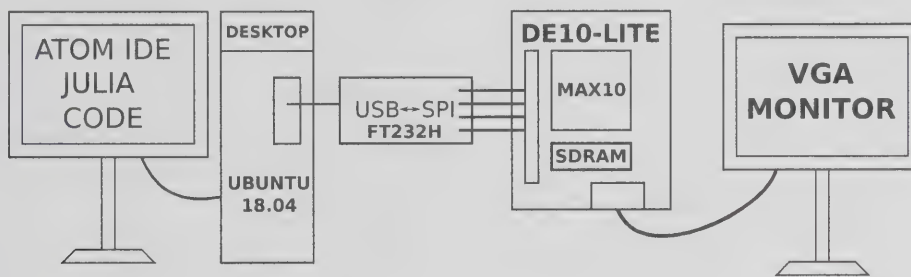
Figure 5: Project System: Desktop -> USB-to-SPI -> FPGA -> VGA Monitor

The Julia code is located in the "src" directory of the github repository linked in the introduction.

9 FPGA and Verilog Books

Notes on books which might be useful to an FPGA beginner.

9.1 Hobbyist FPGA Books

"Designing Video Game Hardware in Verilog" Steven Hugg, first printing 2018

A very practical introduction to digital hardware using the early history of video games as a means of illustrating the technology. The book includes significant introduction to Verilog and its relationship to the physical circuitry.

However, it is not primarily an FPGA book! The reader is encouraged to use a web-based Verilog simulator:

http://8bitworkshop.com

There is an example FPGA which uses the iCE40HX-1K iCEstick. This is a low-cost device ($40 on Amazon). Development can proceed with the official Lattice tool chain, or with an open-source system known as "IceStorm":

http://www.clifford.at/icestorm

"Programming FPGAs, Getting Started in Verilog" Simon Monk, 2017 McGraw-Hill Education

Good coverage of beginning FPGA using the boards Elbert 2, Mojo, and Papilio.

https://www.amazon.com/Programming-FPGAs-Getting-Started-Verilog/dp/125964376X/ref=sr_1_1?keywords=fpga+monk&qid=1564448368&s=books&sr=1-1

9.2   Verilog

You will need to go up the learning curve on Verilog (or VHDL). Here are a couple of inexpensive books (< $20) which will get you going:

"Designing Digital Systems with SystemVerilog" Brent E. Nelson, Brigham Young University

https://www.amazon.com/gp/product/1075968437/ref=ppx_yo_dt_b_asin_title_o00_s03?ie=UTF8&psc=1

"Exploring Digital Logic" George Self.

http://www.lulu.com/shop/george-self/exploring-digital-logic/paperback/product-22747579.html

10   Simulation

The FPGA developer should develop skills in FPGA simulation. The "blinky LED" project I attempted prior to the SPI bus project required me to do that. I had a minor, but persistent bug which brought my work to a halt. Trial and error with the board and re-doing the FPGA got me nowhere fast. After I had a simple simulation running, the problem was quickly resolved. Fortunately the Quartus tool does most of the work to set up the simulation. A small bit of hacking of "do" (TCL) files is required.

I recommend using the resources on the Intel web site to explore the basics of setting up and running simulations via Quartus.

11   Conclusion

I was able to find an FPGA development board and "starter project" that met my requirements for a beginning in FPGA development. A bare minimum of Verilog modifications were required to make the project function correctly.

Most of the FPGA "design" was done using the system-level "Platform Designer" tool.

I was able to add memory-mapped read of the SDRAM and FPGA registers via analysis of the Avalon bus structure. The Julia programming language was used to create and decode Avalon bus transactions. Direct access of the SPI device C shared library from Julia code was used.

The next stage of the project will be a practical ham radio application. I have an antenna rotator which I want to control via wireless. I think it will be possible to create a real-time state machine on the FPGA which will handle the motor drive along with a Hall sensor and counter for positional feedback. Driving the FPGA from the SPI port of an SBC, along with a WIFI connection, should allow the entire control unit to be wireless. Solar powered? Maybe.

Modulation – Demodulation Software Radio

Can Earthquakes change and create Shortwave Propagation?

(A 4-year study of measuring background noise and propagation concludes that this is the case.)

Goups.io user group: https://groups.io/g/MDSRadio

MDSR website: http://users.skynet.be/myspace/mdsr

Alex Schwarz, VE7DXW, alexschwarz@telus.net

1. Intro:

There is well-established scientific evidence (ref 1,2) that earthquakes cause changes in the ionosphere. Experimental results, reported here, support that conclusion by presenting recorded changes in the ambient noise level at shortwave frequencies during an earthquake.

Furthermore, recently published experimental results from Los Alamos National Laboratories (ref 3) for the Cascadia fault, and theoretical results (ref 4) predicting that sound waves carry inertial mass, show that tectonic effects can be detected prior to the release of an earthquake, contrary to the current scientific consensus (ref 5).

The RF-Seismograph recorded such an event on Nov 1st. The spikes and the signal dropouts that were recorded could not come from space, due to solar inactivity (see case study in article). It also caught the eye of the RF-Seismograph team and we went and investigated the phenomena. At the same time, while listening to the local news radio station, it was announced that there had been an earthquake, a M4.9, just north of Vancouver Island! The times of the quake and the measured spike matched in time!

Now, the RF-Seismograph team has been collaborating with USGS to find a correlation between HF propagation and earthquakes. USGS has provided us with a list that contained 171 M6+ earthquakes for the 4 years the RF-Seismograph has been collecting data.

We have recreated the data of the propagation and noise level measurements on days that had a M6+ earthquake and see how much of a change there was visible in our measurements. The findings will be discussed in the following pages.

Using the RF-Seismograph, we are slowly starting to understand more on how the field lines that earthquakes create behave and change propagation. Considering the fact that there are over 1000 quakes (mostly small ones) every day, it becomes quite clear that earthquakes are responsible for a large part of propagation, especially on the lower bands and during solar minimum.

Note: The RF-Seismograph system was developed to measure and collect data on the changes in the ionosphere and radio propagation during the solar eclipse. It uses a shortwave radio and a multi-band vertical antenna to measure background noise and amateur radio transmissions. After the eclipse the team decided to leave the RF-Seismograph running, and now we are in the 4th year of operations. For more info on RF-Seismograph go to: http://users.skynet.be/myspace/mdsr/index.html

1. History of HF Monitoring

To monitor the HF background noise and to intercept radio signals is nothing new. It has been done by the military for a long time. They must have seen the same spikes, noise increases and propagation changes, but could not correlate the events with earthquake times. So most of the noise changes were considered to be unknowable or caused by the sun and ignored.

Fortunately, we now have the Internet. We can now easily call up information on earthquakes with an amazing detail and on a real time basis; thanks to USGS. The Internet made it possible for us to gather all the information and correlate the times of the earthquakes with the reception of signals and the interference caused by them.

Picture of the **AN/FRD-10** HF Monitoring Station, Galeta Island, Panama

Technical information

- **Country of origin:** United States
- **Introduced:** 1961
- **Type: Wullenweber antenna array**
- **Frequency:** Low Band 2 MHz - 9 MHz; High Band 9 MHz - 32 MHz
- **Inner Array Antenna radius:** 393.5 ft (119.9 m)
- **Inner Array Reflector radius:** 366 ft (112 m)
- **Outer Array Antenna radius:** 436.75 ft (133.12 m)
- **Outer Array Reflector radius:** 423.5 ft (129.1 m)
- **Range:** 3,200 nautical miles (5900 km)
- **Antenna Cost (1970)**: $900,000 ($5.81 million today)
- **Electronics Cost (1970):** $20 million ($129 million today)
- By combining multiple stations, a radio signal could be traced to a 100 km^2 area anywhere in the Atlantic or Pacific.

A total of 16 stations were built, 14 in the USA and US bases around the world, and 2 in Canada. The 2 located in Canada and the one in Puerto Rico are still in operation by remote control; the rest have been demolished.

2. How do Earthquakes create Electromagnetic Fields that change propagation?

The science on electromagnetic fields of earthquakes has been discussed for many years and excellent papers and books exist on this subject (see reference at the end). But geologists have never embraced this part of an earthquake, because it is dangerous and cumbersome to collect data on site. Most of their instruments are mechanical and need to be on location in remote, inaccessible parts of the planet to work properly.

The piezoelectric effect and micro-fractures are the main contributors of electromechanical processes during an earthquake. All of them have been confirmed in lab tests as valid physical processes that can create electricity using mechanical energy. When free electrons flow along the path of least resistance field lines are generated. Field lines are a part of electricity and do not exist on their own. Since all the processes must occur before the quake, it will be possible to measure the changes several hours before the quake releases.

How does an earthquake build up before the actual release of mechanical vibration
- Piezoelectric effect of rocks sliding and vibrating on top of each other.
- Micro-fractures of rocks releasing vast amounts of free electrons.
- Electrons move up towards the surface or sea-floor and circulate around the quake area.
- Electromagnetic fields start to emerge out of the earth crust and move upward towards the ionosphere.
- Since the ionosphere contains charged particles, the magnetic field interacts with the ions and creates a hole or a dome of charged particles, affecting radio waves passing through (see graphic below).
- For more information see Scientific American Oct. 2018: "Earthquakes in the Sky" and reference at the end.

3. Distortions of the Ionosphere by Earthquakes

The ionosphere is constantly morphing. Well understood is the impact of the solar wind and solar flux on the ionosphere and the earth's magnetic field. The 24 h day and night and seasonal changes are very well represented in models that we use every day to predict propagation.

The image below shows how magnetic field lines of an earthquake reach into the ionosphere and disturb or bend the layers, breaking existing radio paths or creating new ones. The signals which the RF Seismograph receives drop out, or new connections only last for a few hours while the quake is active.
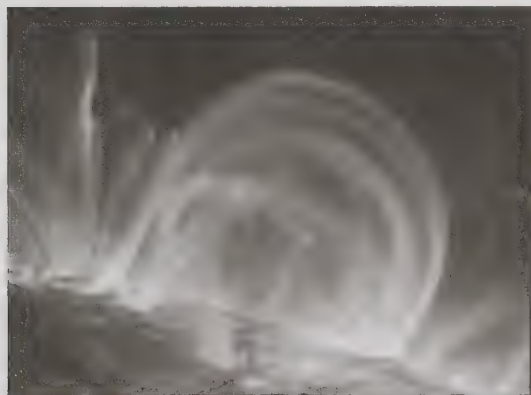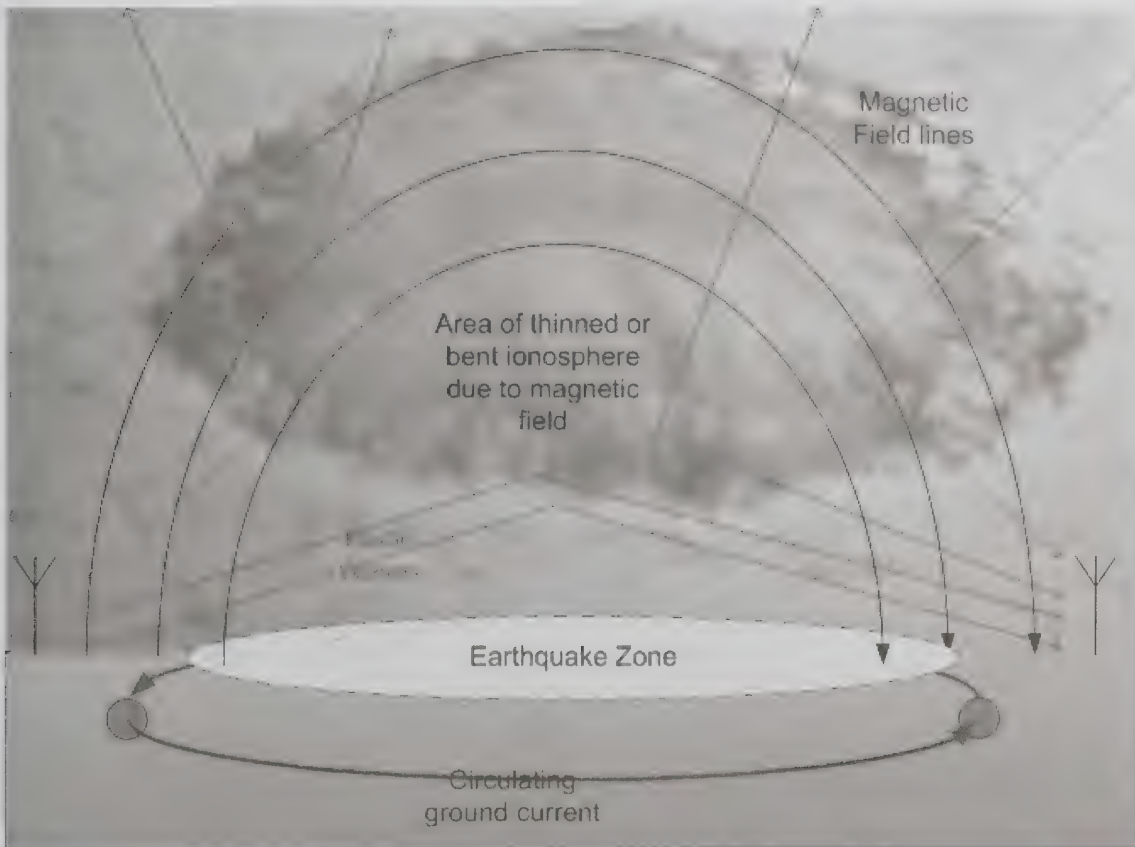
This can be seen as the equivalent of a magnetic field shooting out of the surface of the sun. Because of the hot plasma, the field lines are visible. This process on the sun is much more energetic than an earthquake here on earth, but the physics are the same.

5. How are earthquakes visible on the RF-Seismograph?

The different stages of the quake as seen by the RF-Seismograph
(Case study for M4.9 event, 256 km SW of Pt. Hardy, N-Vancouver Island, BC)

- Energy buildup – noise increases on 80 m (red) starting at 00:00 UTC.
- Disruption of 40 m, 30 m and 20 m bands – communication dropout (lines go flat).
- Quake releases at 04:23 UTC.
- The energy buildup and blackout continues after this quake for the same time the before the quake (2 h) for at total of 4 h.
- After the energy is released, the ionosphere starts to rebuild slowly and normal communication continues.
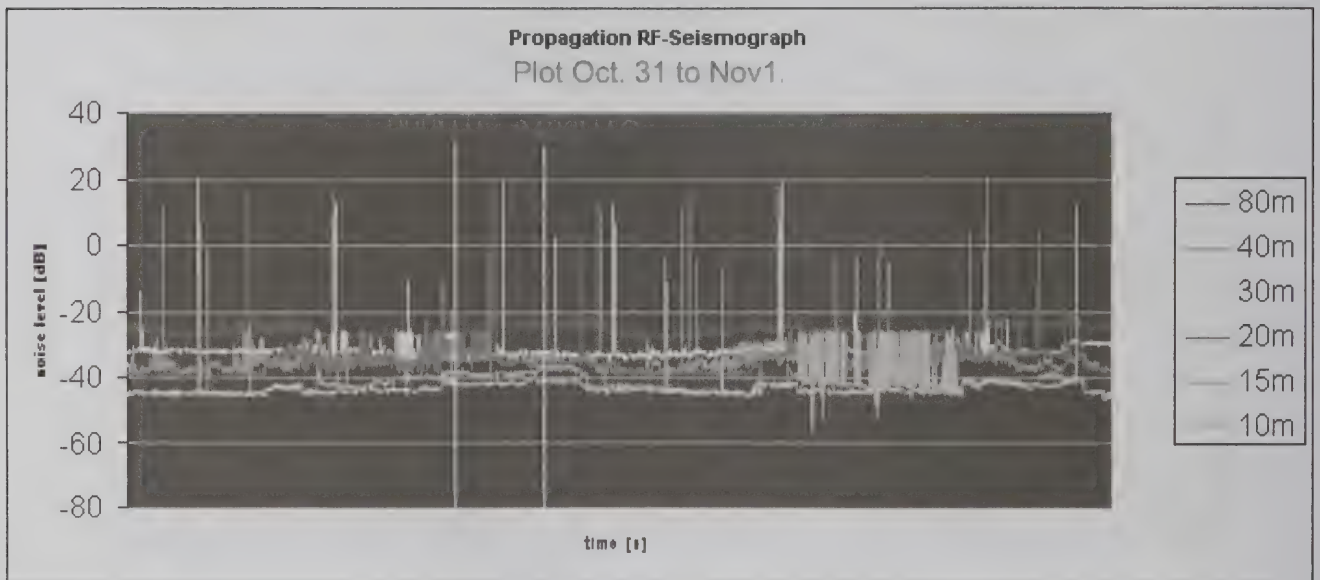
Note: The sharp vertical spikes are static crashes. They can also be produced by breaking field lines.

6. Comparison: RF-Seismograph vs. Ionosonde

The fault line north of Vancouver Island is very active. It releases medium-sized quakes multiple times per year. Usually there is one larger tremor with several aftershocks. Such a series of two earthquakes, within 1 h, have occurred in the Pacific, NW of Vancouver Island on 22nd April 2019. Both of them were picked up by the RF-Seismograph; the first one, on 80 m (top graph, top line) and second one on 40 m (top graph, bottom line). The Ionosonde at Arguello Pt. (lower graph, thick line) picks up the changes as well and even shows the individual quakes as spikes. Even though the Ionosonde and the RF-Seismograph measurements agree in this case, the Ionosonde network does not pick up quakes as easily as the RF-Seismograph.

There are three big difference between the RF-Seismograph and the Ionosonde

- The Ionosonde only measures in one direction vs. the RF-Seismograph, which will pick up any signal from any direction with its omnidirectional multiband antenna. The main focus of the RF-Seismograph antenna is on the horizon, except for 80 m which uses NVIS (Near Vertical Incident Skywave) propagation.
- The data capture time for each frequency is 7 s with and interval of every 52 s. The Ionosonde records changes in the ionosphere only every 15 minutes! The minimum frequency of an earthquake as described in "**Computation of seismograms and atmospheric oscillations**" (see ref. at the end) is 0.00368 and 0.0044 Hz, with two ground periods of 271 and 227 s. Both frequencies are too fast for the scan time of the Ionosonde network.
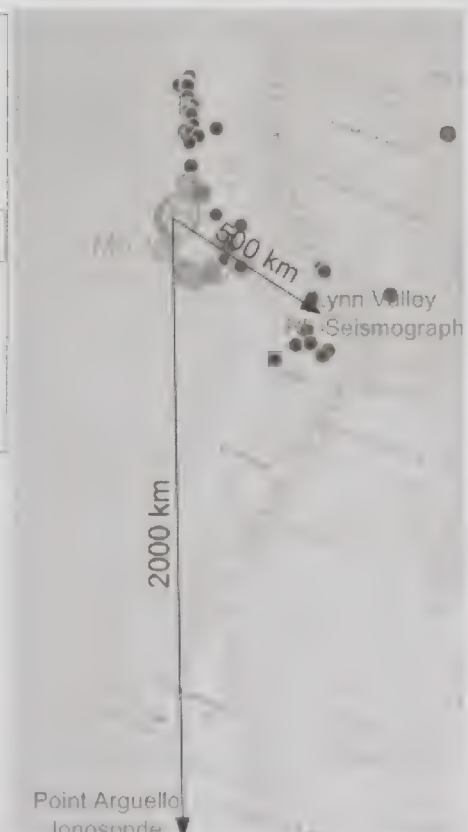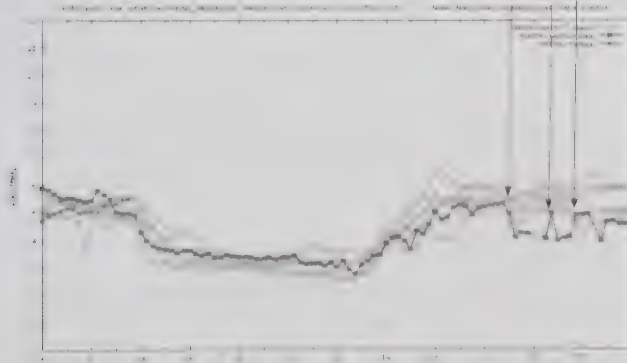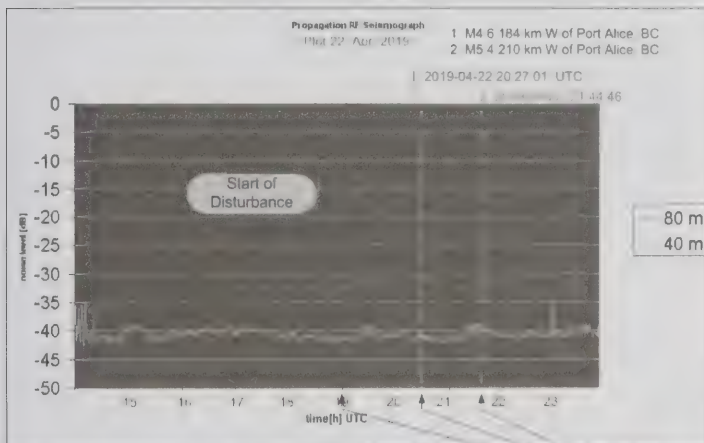
- The RF-Seismograph is passive and listens to all digital amateur traffic on the bands it scans. It works like an oblique Ionosonde with no fixed transmitter. The RF-Seismograph uses RX only and therefore does not need a license to operate.

RF-Seismograph - Lynn Valley (top graph)

- detects a disturbance starting at 19:00 UTC,
- measures a peak on 80 m when the first quake releases
- measures a peak on 40m before second quake

Ionosonde – Pt. Arguello (bottom graph)

- detects a drop in the foF2 frequency at 19:00 UTC
- measures a peak during the first quake
- measures prolonged peak during second quake

Note: one measurement only every 15 minutes

7. Comparison: RF-Seismograph vs. standard Seismograph

The USGS has a website where anyone can create their own seismogram (see ref. for details). This is how we recreated the black full wave graph of the M7.5 event that occurred in Papua New Guinea on 6th of May 2019. The RF-Seismograph also recorded this quake, and matching the two would proof that both were created by the same event.

Propagation RF-Seismograph
Plot 06. May 2019

**M 7.5 - 33km NW of Bulolo,
Papua New Guinea**

Runtime difference: The shockwave of the earthquakes travels in the crust at about 26000 km/h, in the ionosphere the speed is different. Distance between the monitoring station in Vancouver, BC and the epicenter is 10000 km to the seismograph in Berkeley it is 9800 km.
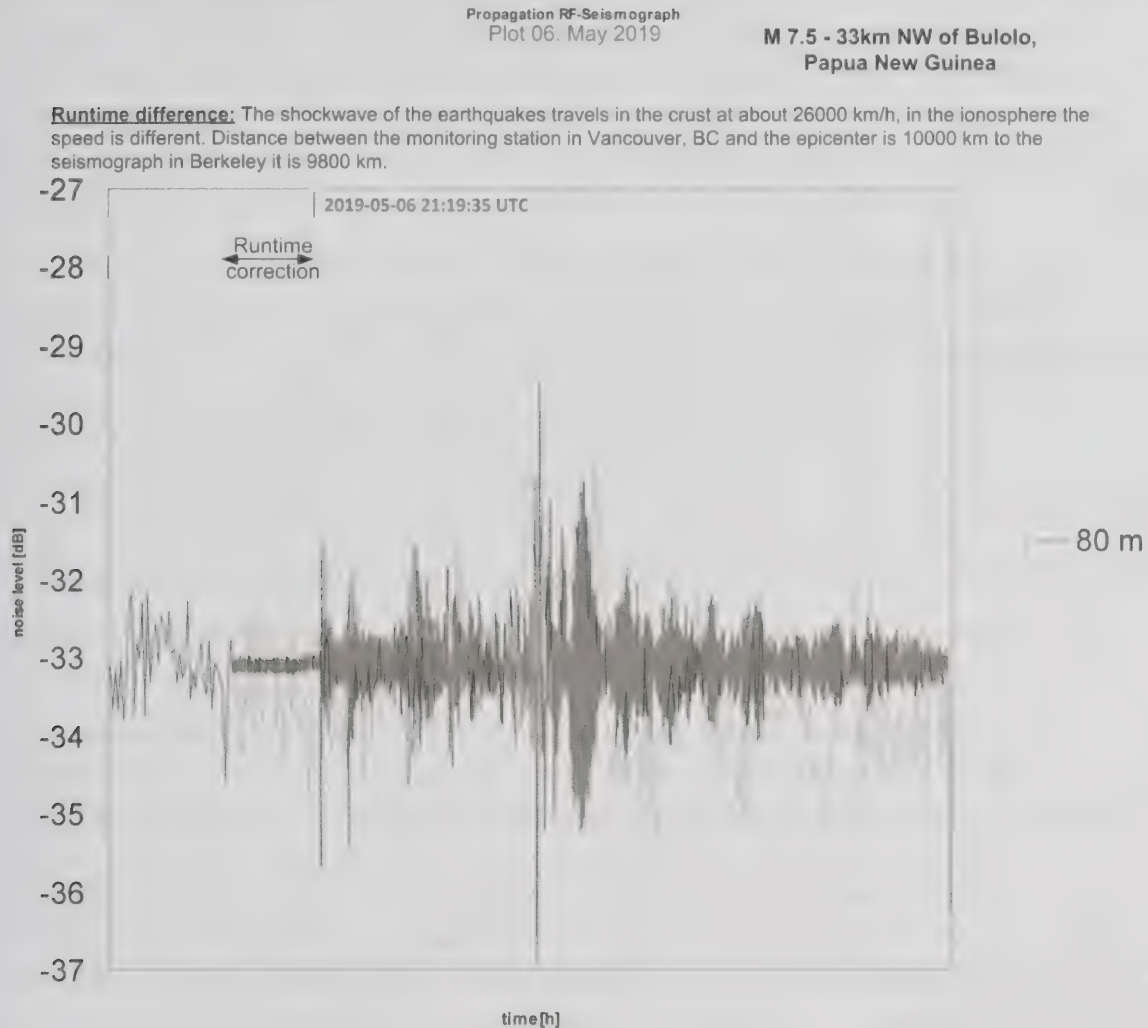
- Earthquakes are very distinct and the wave energy released is like a finger print.
- The black graph is a full wave representation of the physical shock waves recorded by USGS.
- The red graph is the recording of the event on the RF-Seismograph (RMS) on 80 m. The main shockwave energy level and timing match the recording from USGS, confirming that both are caused by the same event.
- Since mantle shockwaves travel at different speed the events arrive at different times at the recording stations.
- Phasing distorts the outer pattern of the signals that travel over 10000 km in different media (crust vs. air).

8. The 4 Year Propagation vs. Earthquake Study

Why is it so difficult to prove that earthquakes have electromagnetic properties?
In previous studies the measurements were done on location, mostly using portable VHF radios with small antennas, and only one specific quake was measured at a time. This makes it very hard to 'catch the quake in the act'. It is also very dangerous and time consuming. If one stands right in the field dome and transmits VHF radio waves, they are at a steep angle to the field lines. The radio waves also penetrate the ionosphere perpendicularly. The ionosphere refracts VHF very poorly, making measurements difficult. In order to actually measure the effect on propagation with the RF-Seismograph, one has to be at least 500 km away from the quake and HF frequencies have to be used. This is necessary because we want to measure the radio signals that bounce back from the disturbed ionosphere.

Why is this study different?
Once we realized the very chaotic nature of earthquakes and linked this to the RF noise they generate (electromagnetic white wideband noise, from 0.0044 Hz into the VHF range), we understood why it is possible to study almost all earthquakes from one location (besides triangulation). By using HF and considering the fact that earthquakes can create RF signals that have several megawatts of power output (possible even more), they are easy to pick up on 80 m. By correlating the time of the quake with the spike on 80 m we can also verify the changes in propagation by the field lines as measured by the RF-Seismograph! The RF-Seismograph combines both events in one graph for easy verification.

- In all, 171 earthquakes were studied: All M6+ events from the beginning of our recording (Aug 2016) to today. Events were provided by USGS, and the quality of the data is high.

- 961 days of recorded data with 171 M6+ quakes amount to one major quake every 5.6 days. Approximately 17.3% of background noise is affected by these strong events. Since we only looked at 6+ events, we can conclude that a lot of the background noise we monitor is also created by smaller seismic events (and there are a lot more of these). If one looks at smaller quakes the (<M3.0) the earth really never stops shaking. There is a lot of energy even in small quakes and they are the major source of the rumble one hears when a HF rig is set to 160m or 80m. It would be interesting to take this experiment to a different planet or moon and see if this is actually the case.

- Only 15 quakes did not have RF noise associated with them.

- 1 day out of 961 was not recoverable due to data loss.

- In 26 cases the time of the disturbance did not match the time stated in the USGS report.

- In 122 quakes (72%) we were able to see a noise increase in the 80 m band either before, after and before and after the quake released. The "before and after" is the most common one. More analysis is needed.

- Introduction and Study of Earthquakes (also see also ref. at the end)
  http://www3.telus.net/public/bc237/MDSR/IntroductionRF-SeismographandEarthqakes.pdf

- The study is still continuing and we need your help to set up more monitoring stations.

9. How to store and search the data we have collected

The data storage is at the moment a simple text file utility. Every day the RF-Seismograph spits out one csv file that has 1661 lines of data. One record = 6 measurements every 52 s. At this moment there are about 1000 individual files. They are uploaded to the group on regular bases for backup and access of group members.

The data is accessible to all IO User group members – membership is free, so please join us! The data files are kept in the files section at our https://groups.io/g/MDSRadio group.

Sample Data:

Timestamp – f [kHz]	3576	7076	10138	14076	21076	28076
06-07-2016 23:56:12 [dB]	-38.59	-49.25	-47.87	-46.16	-51.56	-55.76
06-07-2016 23:57:04 [dB]	-37.61	-50.71	-48.69	-46.3	-51.47	-56.25
06-07-2016 23:57:56 [dB]	-38.21	-50.9	-47.52	-47.35	-51.3	-55.92
06-07-2016 23:58:52 [dB]	-39.3	-50.03	-47.52	-47.27	-50.16	-55.55

Note: The data collection times have deliberately been kept off the common 1 minute markers and are pseudo random to avoid synchronization bias with digital broadcast.

The future storage of data

We would like to see that all the data gets put into a SQL data base. So we are looking for volunteers who are willing to spend some time to get this set up. If you are interested in this please contact us.

Conclusion

Earthquakes are very hard to hear on the radio, but their effect on propagation is undeniable and easy to measure. Shortwave radio operators use the propagation created by earthquakes every day and do not know it! Most of us believe or have been lead to believe that this propagation comes from the sun, which is only a small part of the story. With earthquakes in the picture a new era of propagation research can begin and amateur radio operators are at the edge of new science, again!

Now we come to the big question: Is it possible to predict earthquakes and evacuate people before the event? With the provided measurements it seems that most quakes have a precursor noise level that could be detected and used to alarm a region. It will certainly add another useful tool to the measurements of earthquakes and it will create more certainty that a region is actually getting shaken to a level that causes damage and cost of life. Combined with regular seismographs, it could improve prediction, but clearing an area and then have nothing happen is the worst nightmare of any official; or even worse, after the all clear is given, disaster strikes!

References:

(ref 1) Philippe Lognonne, Eric Clevede, and Hiroo Kanamori, Geophys. J. Int.(1998) 135,338-406

http://www3.telus.net/public/bc237/MDSR/Atmosphere Osc.pdf

(ref 2) Juliette Artru, Thomas Farges and Philippe Lognonne, Geophys. J. Int (2004)158, 1067-1077

(ref 3) **https://twitter.com/LosAlamosNatLab/status/1111386164995321856?s=20**

(ref 4) Angelo Esposito, Rafael Krichvsky,and Alberto Nicolis, Phys. Rev. Lett, 122,084501(2019).

(ref 5) Robert J. Geller, David Jackson, Yan Kagan, and Francesco Mulargia, Science 14 March 1997; Vol 275, Issue 5306 pp. 1616 DOI:10.1126/science.275.5306.1616

Scientific American Oct. 2018: "Earthquakes in the Sky"

http://www.ep.sci.hokudai.ac.jp/~heki/pdf/Scientific_American_Vance2018.pdf

Earthquakes Canada: **http://www.earthquakescanada.ca**

U.S. Geological Survey: **https://www.usgs.gov/**

Sergey Pulinets, Kirill Boyarchuk, Ionospheric Precursors of Earthquakes, ISBN 3-540-20839-9 Springer Berlin Heidelberg New York

Northern California Earthquake Data Center (hosted by the Berkeley Seismo Lab)

http://ncedc.org/bdsn/make_seismogram.html

Access to Study for 2017, 2018 (2019 is part of 2018):

http://www3.telus.net/public/bc237/MDSR/Matches-RF-Seismograph and Seismic data for 2017.pdf

http://www3.telus.net/public/bc237/MDSR/Earthquakes visible with RF-Seismograph 2018.pdf

Support for the RF-Seismograph for Linux and Raspberry Pi: **https://groups.io/g/MDSRadio/**

Download MDSR software for PC from: **http://users.skynet.be/myspace/mdsr/**

Our SciStarter project can be found here: **https://scistarter.org/rf-seismograph**

How to Kill Packet-Radio & APRS? Come to Serbia! (Part 2)

Miroslav "Misko" Skoric, YT7MPB

IEEE Austria Section; NIAR India

email: skoric@ieee.org
packet: YT7MPB@YT7MPB.#NS.SRB.EU

Abstract

In this paper, we continue analyzing pervasive failures in Serbian policymakers' strange decisions, as well as continual country's ham radio leadership's wrongdoings – that all together significantly leads not only to the stagnation in development of VHF & UHF ham radio data infrastructure but also to possible extinction of anything else but contests in telegraphy and 'fox-hunting'.

1. Introduction

In the previous installment of this series, I mentioned that at the time I was writing my first paper for DCC conferences – the paper published in the proceedings of the "22nd Annual ARRL and TAPR Digital Communications Conference" (Skoric, 2003), "I ceased any connection with governing people in Serbian ham organizations for good." (Skoric, 2018). Well, I was wrong: Almost twenty years later I came back to Amateur Radio Union of Vojvodina (Savez radio-amatera Vojvodine, abbr. SRV) to serve as the Union's secretary.

That was not my idea at all, so let me tell you how that happened. Somewhere in November 2018 I received an email from Polish radio amateur Armand, SP3QFE, who asked me whether I knew any ham skilled in ham satellite operations in the area of Sremska Mitrovica city, some 100 kilometers far from my place. He asked me that because there was a lady teacher in an elementary school in Sremska Mitrovica who wanted her school to participate in ARISS contacts with scientists at the International Space Station (ISS). So she managed to locate SP3QFE who served as the regional coordinator-mentor for ARISS, and then he managed to find me as a possible source of information. Unfortunately, I had no previous experience with ham satellites, so I sent an email to the Union's office, asking for help. The person who responded was Stanisa, YU7AC, one of the Union's vice-presidents. We exchanged few emails about the topic, and I eventually paid a visit to the Union's office. After some conversation, YU7AC asked me to serve as another vice-president because they had vacancy for one member in the governing board. Having in mind what I wrote in (Skoric, 2018), I refused his invitation by stating that I did now want to talk about any 'comeback' until I clearly see that the board is doing something to improve the rules related to CEPT status for the 2nd class (the former no-Morse E-category) of Serbian hams. And instead of talking in that direction, he quickly tried to persuade me to take the 1st class license examination. I refused again by stating that I was too busy with my academic activities to waste my time with learning for an exam that "wouldn't prove anything" (self-citation).

After a week or so I received an email from him, telling that in the evening on the same day there was the examination session at the local club and

if I wanted I could enroll for the exam. I thought I could not lose anything by going there, and found my copy of the textbook titled "Priručnik o stručnom osposobljavanju članova Saveza radio-amatera i načinu organizovanja ispita" ("Handbook for educating members of SRS and defining amateur radio examination procedures") written by Djordje, YU1KH, and published 1995 by SRS (Amateur Radio Union of Serbia) in Belgrade. I spent some three hours to scroll through the book pages in trying to refresh my memory for some electronics theory. Eventually, I passed the examination in the evening – although not so perfectly because I had wrong answers to some questions. (The threshold for successful result was around 70% of the question pool.) Besides YU7AC, the other examiner was Mirko, YU7WW, the president of the local radio club "Novi Sad"-YU7BPQ.

A couple of days later, I received another email from YU7AC, asking me to take the Union's vice-president position by mentioning (citation) "... now when we have solved your CEPT issue" (end of citation). As I was in a strong belief that a person can really change (read: improve) any organizational system when he or she is *inside*, I decided to take the risk, and accepted that (voluntary) position. In other words, I felt that after becoming a member of the Union's board I would be in a position to change things.

I was wrong.

2.   "Let us work"

2.1   New secretary

Few days later I received another email from YU7AC, having the Subject: "Secretary". From the mail's content I understood that I was appointed to the secretary position – not for the vice-prez. I knew that it was not the exact part of the previous talks, but I told myself: Ok, let's do it this way.

(According to the Union's bylaw, only the president is elected by the electoral assembly, and then he or she is free to choose his or her team – including the secretary. In this particular case, I have never met the acting president before. That was Laci, YT7DQ, and he obviously agreed when YU7AC nominated myself for an office. It remained questionable, though, why YU7AC had talked to me about the 'vice-prez' office – rather than the sec'y one. Was that incidental or intentional? To understand this, you need to know that the vice-presidents' roles and responsibilities are not clearly defined in the Union's bylaw, and that differs from the secretary's roles that are listed in details.)

So far – so good. At the very beginning I felt I had successfully joined a new team of people who (I thought) were strongly inclined to keep the organization prosper & healthy. In terms of time, it was the early 2019, and a lot of work was in front of the team: Concluding the previous business year, preparing the annual conference of the assembly (held in March), printing & sending membership fee reminders, and so forth. I took over some of those tasks immediately and without questions because I wanted to show my willingness to be a team-player. In parallel with administrative tasks, I offered various organizational improvements because I still felt as a recent outsider who had just jumped to the wagon and who was in a better position to observe the situation from a 'backseat'.

However, it appeared very soon that other members of the board were looking at me as a kind of a 'servant' who was there to satisfy their administrative needs, such as going to the post office or bank, or to seat in front of the computer and type various reports, etc. I do not need to say that I felt uneasy with such kind of (unpaid) 'job', but I told myself: Ok, I was a retiree anyway, and nobody forced me to attend the office everyday, and even when I visited the office I could also think about some more improvements, and I could put those ideas on my agenda, and …

To examine the 'public opinion' regarding the 'state of the Union', I subscribed to the "YU0V" member mailing list, and posted some inquiries.

My posts were related to reviving packet-radio network on the geographical part of Serbia that was under the Union's responsibility. And I received only few responses – although off-list. (Those who responded did not want to go publicly for whatever reason.) Some responders were skeptical that many (if any) things could be improved in the Union's practices. Their explanation was that the old-fashioned way of thinking within and around the organization was the greatest barrier to any improvement. In other words, the governing people and the majority of members had deep grassroots in a (mis)belief that only 'fundamental' amateur radio modes & activities, such as CW and SSB contesting, were the ones worth to play with. Other activities, such as amateur radio satellites, packet-radio, APRS™, ARISS contacts, and many others, were either unknown to the majority of local hams or totally marginalized by the Union's leadership.

I considered that as a long-term bad politics, and I wanted to change that. So I took an opportunity to rewrite the Union's 2019 yearly plan, by adding activities related to developing packet-radio and APRS™ networks, as well as initiating communications with Amateur Radio Union of Serbia (Savez radio-amatera Srbije, abbr. SRS) in Belgrade, the national capital. The planned communications with SRS would include talks about improving the rules that govern amateur radio in Serbia – more precisely the necessity to accept pan-European CEPT recommendations in its entirety. As discussed in (Skoric, 2018), Serbia has accepted only parts of recommendations that favored telegraphers, and I argued that no ham category should be ever marginalized as a result of preferred operating mode(s).

And by the way, even though the 1st amateur radio license in Serbia was listed as the CEPT-compliant in *ero.dk* website, in the license itself it was not clearly noted that the licensee had a fully-compatible CEPT document, see Figure 1. In fact, it would be more suitable for license holders in the 1st class (Figure 1, left) that their 'plastics' include some words like "CEPT-compatible", or like. Otherwise, some foreign administrations may require additional paperwork when a licensee travels abroad.
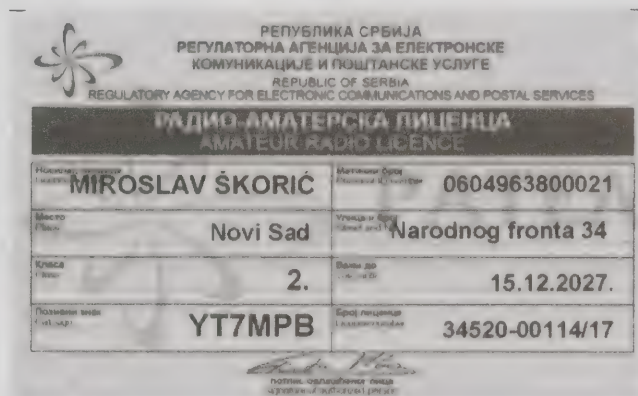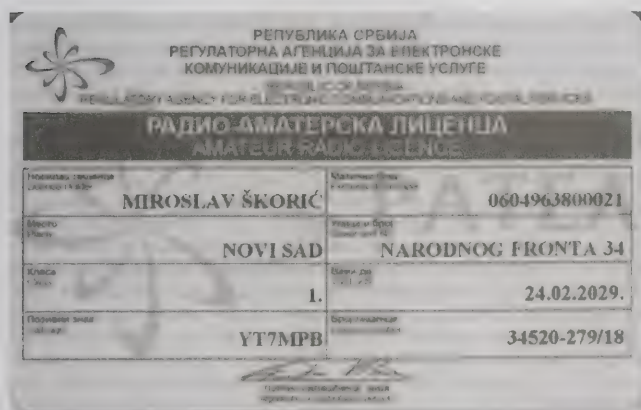
Figure 1. Ham radio licenses of the 1st class (left) and the 2nd class (right).

Not to mention that it would be also suitable for license holders in the 2nd class (Figure 1, right) that they obtain "CEPT Novice -compatible" status and be labeled accordingly. That was also a part of my agenda because I noticed that many licensees in the 2nd class spend their summer or winter holidays in neighboring countries, and they should be allowed to use their handy or vehicle VHF/UHF amateur radios while travel around. By the way, it is quite usual today to see on the APRS™ maps foreign operators traveling throughout Serbia and reporting their positions without any special requirement. Having in mind that APRS™ in this part of Europe is almost

exclusively used on VHF band (144.800 MHz), and that the majority of license holders in the 2nd class use handy VHF/UHF radios, I see no reason why to keep them legally unable to use their stations abroad for making friendly ham contacts on a short-term basis. Do we really want our VHF/UHF bands to remain fully utilized, or not? To illustrate international road traffic in Serbia, I give an example of a Turkish amateur radio operator TB2BCE who frequently travels by Serbian highways as a truck driver (Figure 2).
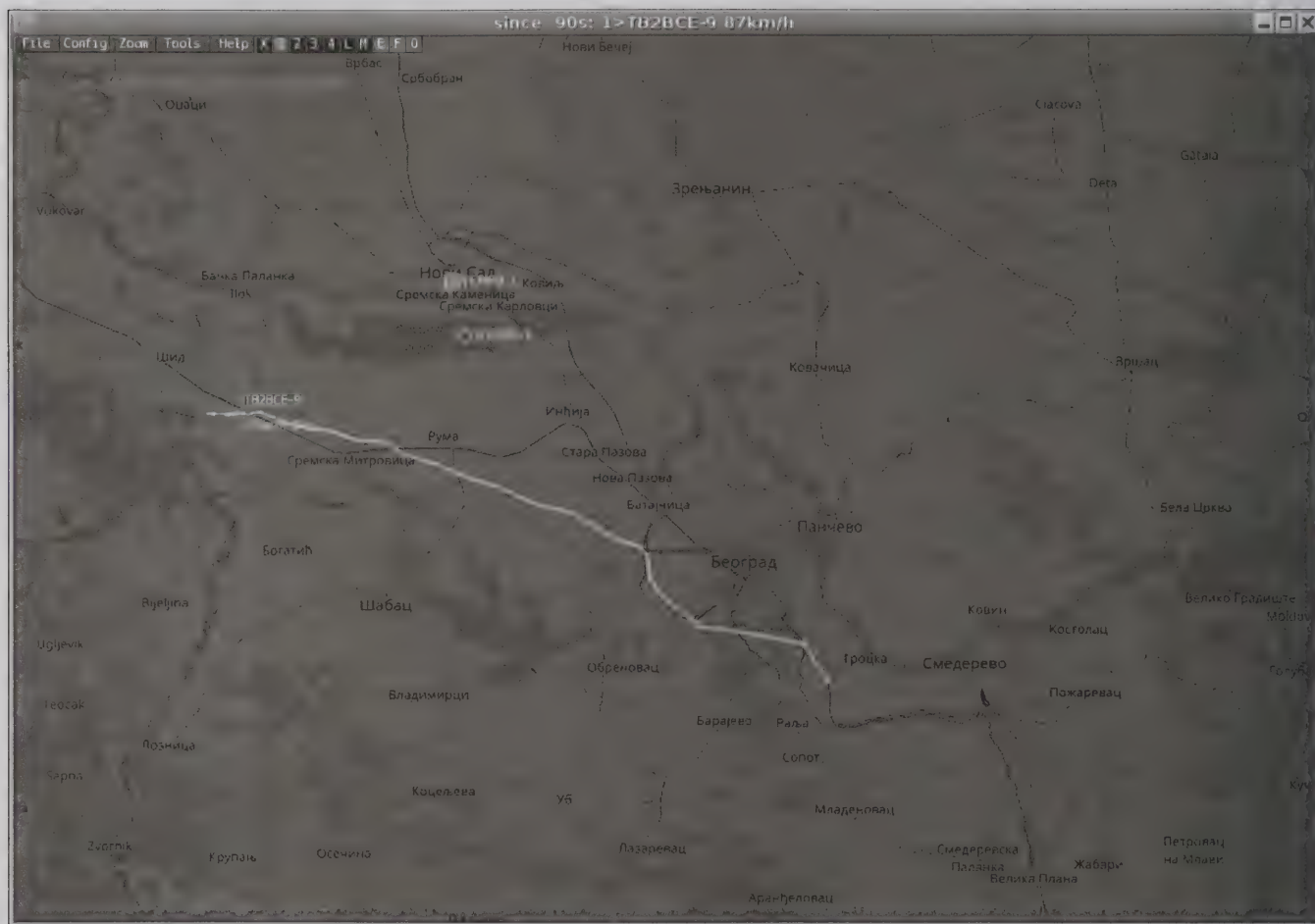
Figure 2. Turkish amateur radio operator TB2BCE who traveled from east to west on January 22, 2018.

There are other examples, such as an Austrian motorcyclist (OE8NDR, Figure 3), or a German car driver (DL1ZM, Figure 4).

2.2   No real political will

Although the president and vice-presidents of SRV did not make visible objections to my proposal for improving the regulations, it appeared that it was easier said than done. When I talked to YU7AC about the proposal, he wanted to assure me that I will have the strongest opponent in his 'good friend' Sinisa, YU1RA, who was, by the way, one of the creators of actual regulations (brought to power in 2011) that decreased ham radio opportunities for Class 2 licensees in Serbia. It appeared that YU1RA had advocated that Serbian Class 2 licensees have not deserved to be included into the CEPT licensing system. YU1RA, who also serves the office of a secretary general in SRS, falsely claimed that the quality of education in the former E-category (now Class 2) was 'so bad' by means that the level of technical knowledge in Serbian no-Morse operators was

'far lower than the standards prescribed by CEPT and/or CEPT Novice practices'. It is really bizarre for the ham society leaderships to express such nonsense because it would seem that they care about wellbeing of the EU more than the EU's own telecommunication regulators. There is an old adage here in Balkans that says something about those (wrongdoers): "They want to be better catholics than Pope".

Unfortunately, the reality is quite the opposite: The SRS leadership does not care about the EU and CEPT regulations at all. Instead, by falsely claiming that they 'protect' pan-European amateur radio traffic from apparently/allegedly 'unskilled & uneducated' (no-Morse) operators (Class 2) in Serbia, they actually diminish chances that this country modernizes itself in many areas, including ham radio technologies.
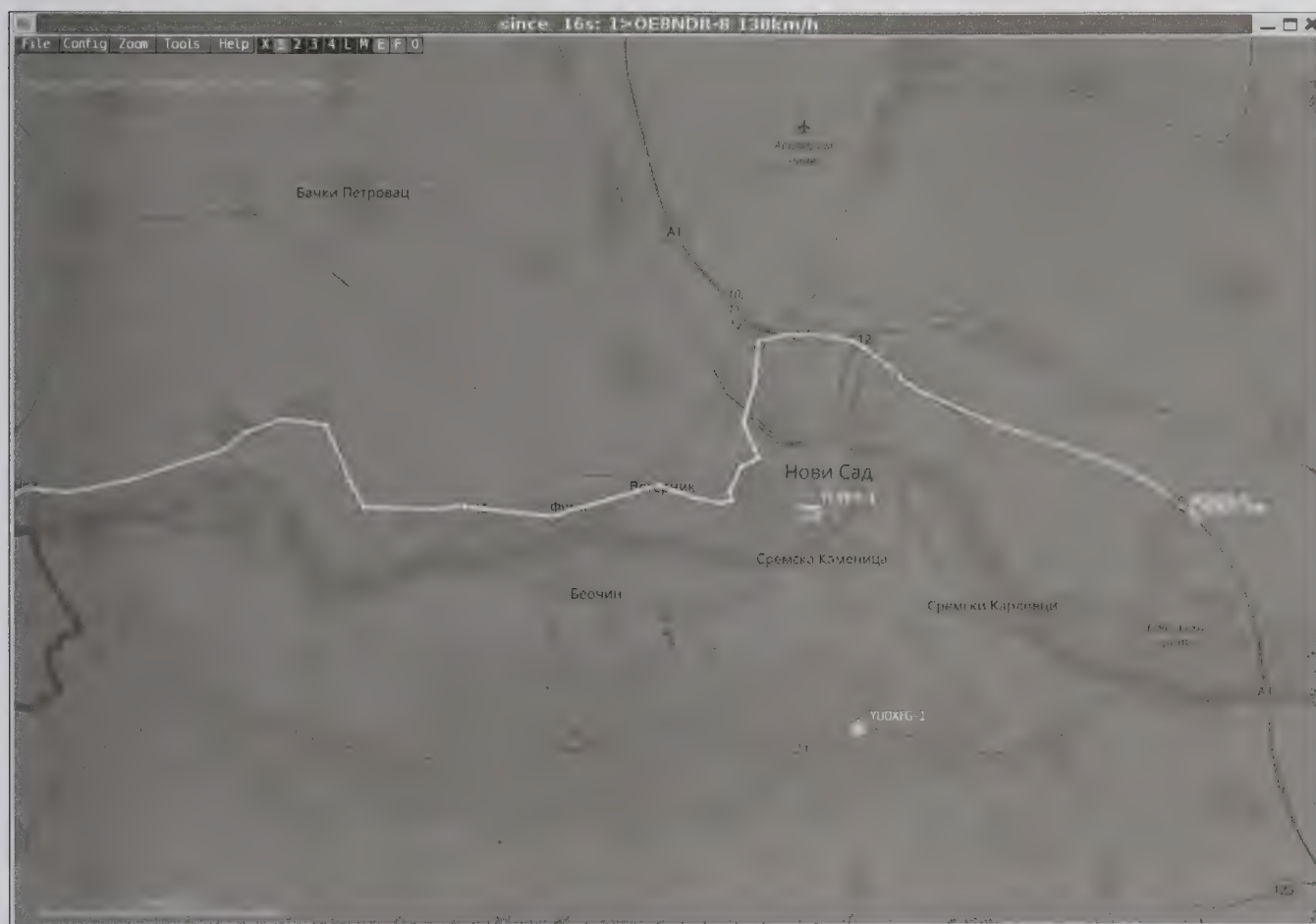
Figure 3. Austrian amateur radio operator OE8NDR who traveled from west to east on April 19, 2019.

2.3   I say, you say ...

Here I list some examples that are witnessing bad attitude in Serbian ham leaderships when it comes to modernization & reforming ham communities in the country. It became common among the ham radio traditionalists in Serbia to urge candidates to 'avoid by any means' the examination for Class 2 (former no-Morse category), as well as to 'skip' the 'unnecessary Class 3' (the beginners category). Recently I was in the local club YU7BPQ when one of the older members Ljubisa, YU7BG, strongly suggested to a newcomer to go 'directly to Class 1' - even though the young candidate clearly expressed his interest to start with handy VHF/UHF radios. Unfortunately, YU7BG was resolute: "Handy? No way! That's not ham radio! Come here to the

ham shack to see what telegraphy is." I was amazed by such level of discouraging prospective candidate to start from basic things. (Not to mention that just few days later YU7BG asked the club's executive board to approve his request for purchasing new batteries for the club's handy talkies. He did not bother to explain the reason for purchasing batteries, and did not clarify his intention for using portable radios at that time.)

Figure 4. German amateur radio operator DL1ZM who traveled from north to south on December 23, 2017.

Even though I made it clear that my reactivation in the SRV executive board was motivated primarily to remove aforementioned bad attitude among 'ham fundamentalists' towards no-Morse hams and beginner categories, YU7AC told me in person that "only real hams are those who practice CW and SSB". Really amazing. He knew very well that I was, for many years, the no-Morse ham operator who practiced data modes such as administering packet-radio nodes and AX.25 mailboxes, who kept alive local APRS activities, and so on. However, he was not able to realize that his statement sounded offensive. Furthermore, when I advocated to reaffirm regional packet-radio network, he quickly claimed that "packet-radio was dead" as a ham activity. In opposite to his claim, I recently received quite positive opinions on the actual status of packet-radio in the USA (for example). Brian, N1URO, wrote that he, as the re-elected president of the 'EastNet Packet Network', can confirm many activities in building new nodes in his part of the country (Brian, 2019). David, KI6ZHD, said that "... Winlink email packet,

EmComm used packet, and Classic keyboard to keyboard packet is doing decently well here in the Northern California, USA. There are still a few packet BBSes, lots of TNC pBBSes, etc." (David, 2019). Doug, N7BFS, noted that both Washington and Oregon states in the US had a number of nodes still active, mostly with Winlink/Airmail), and that he maintained four of them in eastern Washington and coordinated with a number of amateurs in the outer areas (Doug, 2019). Joe, AG6QO, pointed out that AX.25 packet is alive and strong in the USA, especially where he lives in the Pacific West:

> Many BBS, nodes, keyboard-keyboard nets, and WinLink gateways using packet. I can node-hop from northern California into Washington state and Canada to the north, and to Arizona, Nevada and Mexico, all on VHF/UHF AX25. Very reliable, proven, and cheap, readily available hardware and software. (Joe, 2019)

Besides the US packeteers, Lorenzo, IW3HER, sent me an email informing on few 'Super Vozelj' nodes that were still active in his area of northeastern Italy, and have been maintained by Cristiano, IZ3LSV (Lorenzo, 2019). However, Rick, KG4OFO, argued "... that AX.25's lower overhead makes more efficient use of RF bandwidth as long as it's not just encapsulating IP" (Rick, 2019), while Tom, VK7NTK, saw difficulties in having more users in southern Tasmania (Tom, 2019). Mike, AB3AP, noticed a lack of packet stations in southeastern Pennsylvania where he lives. He added that there were a few but they're just outside his ability to transmit. Sometimes he can receive them, but they rarely hear him (Mike, 2019).

So this short survey proved that packet-radio was far from 'death' - although experiencing various difficulties in different areas.

3.   More issues

Political discrepancies within Serbia, which started alongside with the fall of former Yugoslavia, have reflected to the amateur radio community as well. For example, Serbian regulatory authority RATEL looks at SRS as the representative of ham radio interests in the country. In the previous installment I discussed about RATEL's blindly following wrong suggestions originated in SRS. Unfortunately, it seems that the problem is not only in the offices of SRS, but also in SRV too. In fact, SRV executive board members tend to keep 'calm & friendly' when it comes to mutual relationships between the two unions. And both sides tend not to make troubles in between after they managed to 'share the territory'. Now SRV handles license applications in Vojvodina province of Serbia, while SRS does the same in the rest of the country. The problem I noticed as the secretary of SRV is that SRV is not able (or not willingly) to properly articulate the problems with CEPT harmonization in front of the SRS leadership, and consequently in front of RATEL as the regulatory authority. And as long as status quo is maintained, I cannot predict much progress in domestic regulations (if any).

For example, although he does not want to say it loudly, it is obvious that YU7AC is not likely to produce any turbulences that, in turn, may produce some negative feelings in both SRS and RATEL. Why? It is unclear, but it is widely known that at least two vice-presidents of SRV (incl. YU7AC) have strong motivation to keep businesses of their private companies well and healthy. Their companies work in areas of telecommunications, so the owners obviously want to stay 'at mercy' of national regulators. Only God knows whether that situation shall be described as a conflict of interest, or not. You can tell me how you deal with similar cases in the US or elsewhere.

Somewhere in 2018, several months before I started to serve as the Union's secretary, there was a dispute between a former SRV vice-president Sasa, YT3H, and the SRS executive board in Belgrade, where YT3H was also one of the vice-presidents in the same time. (It was a

kind of a 'delegating' system where the SRV executive board had a representative in its SRS counterpart, to coordinate activities in both societies.) But, in some incidental situation the SRS president 'fired' that SRV representative, as a result of some personal disputes, or whatever. At that time, YT3H was a member of SRV executive board in full capacity, and 'thankfully' to that fact, SRV prepared an 'objection note' to SRS, related to apparently 'unconstitutional expelling' YT3H from the SRS board. However, few months later, YT3H decided to step down from the vice-president's position in SRV, because of some personal reasons (or, this time, a dispute with the SRV board members). Interestingly, YU7AC 'warmly welcomed' the new situation and canceled sending the objection note to SRS – even though they previously designated expelling YT3H from the SRS board as 'unconstitutional'. I asked myself: Where the ethics disappeared?

Another example is the Friedrichshafen show in Germany. As you know, it is a kind of 'European Dayton' ham meeting. Serbia, as a country, has never had a union's booth there, but some local ham individuals regularly attended the show, by sponsoring their travel and stay by their own personal budgets. However, few months ago I was present when YU7AC had a telephone conversation and told his correspondent that SRS 'appointed' him as a 'delegation chief' for Friedrichshafen – although he was not going to attend the SRS booth at all, and commented that somebody else shall be found to serve at the booth. He also mentioned that part of the costs of travel & stay would be covered by SRS funds (read: membership money). I dared to ask YU7AC what was the purpose of the planned travel to Friedrichshafen – having in mind that the booth never existed and no ham saw that as a big problem. He responded that "... all the countries in the region of Balkans had their booths in previous shows, so that it must be some usefulness of being there ..." Nevertheless, I also witnessed when YU7AC discussed the planned travel with Mirko, YU7WW (the president of YU7BPQ club) and Sloba, YU7HI (the president of YU7BPQ advisory board).

From their discussion, it was clear that the main reasons for going to Germany included some of their private businesses. In general, that's fine – but only as long as they do not pretend to 'represent' Serbian hams there (and do that at the cost of the hams' membership money!)

3.1 Strange budget inputs

According to the SRV's 'constitution', this is a society of ham clubs in the province of Vojvodina (in the northern part of Serbia). In general, those clubs have decided to join SRV (i.e. to form the society on the provincial level). Particular club names are not listed in the 'constitution' because new clubs can appear and some old ones can disappear from the scene. Individual club members are not mentioned in that document at all, while individual 'direct' SRV membership is mentioned as an option for hams who do not have nearby clubs (or just do not want to join local clubs). From that point of view, it would seem logical that the clubs pay the highest membership fee to SRV, followed by those clubs' members (who should pay a little bit less because they also finance those clubs), followed by the individual SRV members (who should pay the least because many of them live outside urban city zones, and hence do not enjoy SRV services/privileges intended for the clubs). But, that's not the case: The clubs pay the lowest fee (1000 RSD per year); club members pay double (2000 RSD), and individual SRV members pay the highest (3000 RSD). Why? Nobody knows. However, that does not seem fair because most of the clubs are already equipped with radios, antennas, and other ham gear, so they make new purchases infrequently. Furthermore, the clubs' members use that gear regularly, so they can save their personal funds. In opposite, individual ('unconnected') SRV members have to finance their own gear *and* to pay the most to the Union. Where's the logic there? And what happens: As a secretary I realized that the majority of those individual hams do not pay their membership fee for several years. Usually, they only pay for the membership once in ten years when they need to renew their

licenses, and only then the unions use the chance to collect some money from them. (In fact, hams in Serbia are not obliged to be the unions' members any more.) From their point of view, those individual ('direct') members look at the membership fee as a money extortion, and because of that they tend to avoid paying for the full decade later. And although SRS and SRV bosses keep claiming that their organize 'important international contests & meetings', the majority of individual hams do not see much usefulness & services of those unions. As a result, instead of circa 500 new potential individual members, SRV manages to collect not more than 80-100 memberships per year.

3.2   No efforts to educate themselves

After my voluntary 20+ years of self-isolation from any organizational ham activity, I expected to see some 'new blood' within the Serbian ham community. I expected to see various efforts for getting closer to the education, science, and research sectors. For example, here in Vojvodina province we have few academic centers (Novi Sad, Sombor, Subotica, Zrenjanin). In all those cities we have active ham clubs. So I expected to see efforts on both local (city) levels and on the provincial (SRV) level, to establish long-term relationships in between the ham clubs and local schools and universities. It is known that amateur radio hardware and software always need modernization and improvements in quality of signal processing, speed and reliability of ham software, and so on. It would be logical to see initiatives for including ham radio in schools' activities, on mutual behalf. (By the way, collaboration with educational sector has been already listed in the Union's 'constitution' as one of the most important tasks.)

Unfortunately, there are no such efforts on the horizon. A former ham club at University of Novi Sad is 'dead' for a long time now. I asked myself why, until I realized that in the past the 'ham fundamentalists' only wanted to use high academic buildings as a place for erecting antennas – antennas for CW contesting, of course. And whenever they erected such antennas, they fed them with so strong power – strong enough to produce interference to the academic lab equipment. As a result, many such falsely 'collaboration' with science & research ended by expelling hams from the university.

3.3   Living in the past

On March 24, 2019, ham leaderships of SRS and SRV 'contested' each-other in patriotism. That particular day was a remembrance day of starting NATO bombardment over the former Yugoslavia in Spring 1999. All the TV stations – particularly those under control of political authorities, celebrated a 'victory over NATO criminals'. Of course, not a single TV station dared to analyze what led to the confrontation with NATO. Nevertheless, YU7AC was hosted by some local media where he pointed out the important role of Serbian hams during the "NATO aggression". Similar content was heard from Dusan, YU1EA, one of the SRS leaders, who was the guest-speaker in the morning program of the state-governed RTS public service[1]. Although there is no doubt that Serbian hams did a lot during the mentioned conflict, it was really sad to hear two unions' leaders so over-motivated to discuss one of the darkest periods in Serbian history.

To make things really bizarre, the same week we held in Novi Sad city the annual assembly of SRV, where the president of SRS informed the audience that the national amateur radio organization had to pay a significant amount of money to the state-owned company that coordinates TV towers in the country. As you know, high TV towers are often used as good locations for positioning ham radio repeaters. Although I am unsure how that area is regulated in the USA and other countries, it seems that Serbian government found a way to make money from those who were supposed to be (and who still believe they are) 'patriots on the first line of defense'.

---

[1] https://vimeo.com/326136500

3.4 Numbers do not lie

On Saturday, May 11, 2019, the annual assembly of SRS was held in Belgrade, the Serbian capital. Approximately one week before the meeting we received the SRS annual report. Among several intriguing parts of it, there was a detail that took my specific attention: The statistical summary of ham examinations per ham clubs, unions, and amateur radio classes. The summary is copied in Figure 5.

Одржани испити по клубовима (само СРС):

Клуб	2018			
	1. кл.	2. кл.	3. кл.	CW
YU1ACR			4	
YU1ADO	1	14		
YU1AFV	2		4	
YU1AKV	3	1	3	1
YU1AXY			5	
YU1BBV	4		6	
YU1FJK			20	
YU1HFG	6	2	7	
YU1INO	1		8	3
YU1SRS	2		5	
Укупно:	19	17	62	4

СРВ испити:

Клуб	2018			
	1. кл.	2. кл.	3. кл.	CW
YU7AOP	12			1
YU7BPQ	10		3	
YU7JDE	2			
YU7KMN	3		6	
Укупно:	27		9	1

Figure 5. Summarized report on ham examinations in between the two annual assemblies (Apr 2018 - Apr 2019).

The upper part of the table shows the number of examinations in ham clubs under the 'jurisdiction' of SRS only (performed on the state territory called "Inner Serbia" - without autonomous provinces of Vojvodina and Kosovo*), while the lower part has numbers for SRV union only. Although it was clear that overall interest in telegraphy exams was marginal (4 in SRS, 1 in SRV, see the fifth column), something else was very strange: SRV did not have a single one Class 2 /former no-Morse, now no-CEPT category/ examination (third column), and the number of candidates for the top-level Class 1 (second column) tripled the number of candidates for the entry-level Class 3 (fourth column). The only conclusion is that SRV leadership worked very hard in 2018 to build up (artificially) the score for allegedly great number of newcomers in the 'top-level' ham group, and they did that (intentionally) at costs of Class 2 and Class 3. Especially at the cost of Class 2 that was slighted for decades, and is still undervalued by self-proclaimed ham 'elites' in Serbia. Amazing! In opposite, the numbers per classes in SRS seemed more logical (the most newcomers were in the entry-level category – 62, where the test was easiest, while Class 2 and the top-level Class 1 numbers were relatively balanced).

So, what can we say after looking at those numbers? Well, although there is no doubt that SRS, as the main national-wide ham group, has always had its 'teething troubles', it is obvious that the SRV leadership was not immune from delusion, obfuscation, lies, and continual tries to make things up. It is unclear how IARU Region 1 office has come to a decision to let SRS/SRV administrations to host next IARU R1 Conference in 2020. There are already rumors that the most important aspect of that event is big money that will come from the IARU budget. (Coincidentally, few days ago in early May 2019, there was an EU/EBRD economic meeting in Sarajevo, Bosnia and Herzegovina, where a BBC-titled person told the audience that Serbia had received millions of € to improve its judicial system, but instead of an improvement the country's position was lowered from some 60th to some 90th place in a global ranking.)

To learn more about the 'harmonization' with CEPT regulations in surrounding countries, I contacted offices of ham unions in Croatia, Slovenia, and Bosnia and Herzegovina. While the first two former ex-Yugoslav republics did not bother to respond to my inquiry, I received a friendly email from Mesud, E75FM, the president of "ARA u BIH" (Bosnian ham association). The content of his mail made it clear that Bosnian ham leadership wanted to be fair to all previous amateur categories, which means they did not want to exclude anybody

from international ham activities. So, they decided to put previous 'upper' classes A, B, and C into CEPT1, and 'lower' classes D, E, and F into CEPT2 (Mesud, 2019). Furthermore, CEPT1 is the Bosnian national equvalent to CEPT Amateur License, while CEPT2 is equivalent to CEPT Novice License. Fair? I think so.

3.5   Is there light at the end of the tunnel?

After realizing so many bad practices within the Serbian ham community, I initiated aforementioned technical works, such as those related to reviving packet-radio and APRS networks. I did it publicly by using the union's mailing list. Only a few responses returned back – most of which pretty skeptical on any positive development. However I will stay optimistic until the bureaucratic and 'ham fundamentalist' obstacles become too much burden.

I also continued to talk with Serbian governmental offices: After having found that Prof. Dr. Irini Reljin, deputy minister in Serbian ministry for telecommunications, tourism, and trade (TTT), has been in charge for sector of electronic communications and postal traffic, I emailed her office few times in April and May 2019. In those mails, I informed her on the issues discussed in this study. No responses from her or anyone else in TTT ministry yet. By the way, I have recently noticed that in June 2018 RATEL closed the public opinions on then announced draft of a new amateur radio rulebook. Unfortunately, I missed that opportunity to act on time. However, although it was predicted that the new rulebook should have been brought to power in mid-2018, that did not happen yet. So I still hope that my suggestions sent to Ms. Reljin would be of use.

Nevertheless, bad feelings are still everywhere. I have recently witnessed when YU7AC loudly diminished the strength and influence of citizens' protests on the streets of Serbia against the actual political establishment, by giving negative comments on the protestants' allegedly low numbers. That partially says something about (but not only) YU7AC's tendency to blindly follow & support Mr. Vucich's rigid political regime that is just a worse copy of the former Milosevich's clique. And yes, I can understand that, for example, YU7AC has business interests in the actual regime, as he had similar interests in the previous governments. But, if that goes at the cost of the wellbeing of radio amateurs, I will be a strong opponent.

4.   Conclusion

As I said at the end of the previous installment, I respect all amateur radio communication modes, including CW and SSB. I am not against them and I never was. Furthermore, I respect them as most traditional and proved ones. However, anyone's skill in practicing them cannot be – by any means – an excuse for neglecting modern ways of computer-related communications, including various data modes. With that idea, I hope that DCC 2019 participants will be in a position to offer new perspectives for improving this fine hobby. In that direction I invite again radio amateurs from the USA (and elsewhere) to consider joining me in preparing materials for new book chapters and slides for technical presentations in developing parts of world.

5.   References

Brian N1URO (2019). *Re: OT: Global AX.25 status?* Personal communication.

David KI6ZHD (2019). *Re: OT: Global AX.25 status?* Personal communication.

Doug N7BFS (2019). *Re: OT: Global AX.25 status?* Personal communication.

Lorenzo IW3HER (2019). *Re: OT: Global AX.25 status?* Personal communication.

Joe AG6QO (2019). *Re: OT: Global AX.25 status?* Personal communication.

Mesud E75FM (2019). *RE: Radio-amateri u regionu (BiH)* Personal communication.

Mike AB3AP (2019). *Re: OT: Global AX.25 status?* Personal communication.

Rick KG4OFO (2019). *Re: OT: Global AX.25 status?* Personal communication.

Skoric, M. (2018). How to Kill Packet-Radio & APRS? Come to Serbia! In *Proceedings of 37th ARRL and TAPR Digital Communications Conference* (ISBN 978-1-62595-101-4, pp. 79-89). Newington, CT: American Radio Relay League.

Skoric, M. (2016). Adaptation of Winlink 2000 Emergency Amateur Radio Email Network to a VHF Packet Radio Infrastructure. In El Oualkadi, A., & Zbitou, J. (Eds.), *Handbook of Research on Advanced Trends in Microwave and Communication Engineering* (pp. 498–528). Hershey, PA USA: IGI Global.

Skoric, M. (2014). Software in amateur packet radio communications and networking. In Matin, M. (Ed.), *Handbook of Research on Progressive Trends in Wireless Communications and Networking* (pp. 122–188). Hershey, PA USA: IGI Global.

Skoric, M. (2013). Security in amateur packet radio networks. In Khan, S., & Pathan S. (Eds.), *Wireless Networks and Security: Issues, Challenges and Research Trends (pp. 1–47). Berlin - Heidelberg, Germany: Springer.*

Skoric, M. (2012). Simulation in amateur packet radio networks. In Al-Bahadili, H. (Ed.), *Simulation in Computer Network Design and Modeling: Use and Analysis* (pp. 216–256). Hershey, PA: IGI Global.

Skoric, M. (2009). Amateur radio in education. In Song, H., & Kidd, T. (Eds.), *Handbook of Researchon Human Performance and Instructional Technology* (pp. 223–245). Hershey, PA: IGI Global.

Skoric, M. (2003). Legal rules and regulations in the amateur radio computer networks. In *Proceedings of 22nd ARRL and TAPR Digital Communications Conference* (ISBN 0-87259-908-6, pp. 215-221). Newington, CT: American Radio Relay League.

Tom VK7NTK (2019). *Re: OT: Global AX.25 status?* Personal communication.

GPS Watch Technology

Darryl Smith, VK2TDS
TAPR Board Member.
vk2tds@tapr.org

From what I can work out, I am a bit unusual in the ham radio world. In addition to being active in amateur radio, and being on the board of TAPR, I am a mad keen runner. I started running about four years back, and have never looked back.

After doing my first marathon[1] at the end of last year (2018), I was looking for a new challenge, and decided to try trail running. Whereas most running events are on roads or paths, trail running involves running in the great outdoors away from civilization. And without the limitations of where cars can drive, trail runs tend to be hillier.

You might have worked out from my callsign that I don't live in the USA. I actually live about an hour south-west of Sydney, in Australia. This is an amazing part of the world, and we are blessed with some amazing places to go running. My first marathon started by running over the Sydney Harbour Bridge, and finished at the Sydney Opera House.

But there are some other amazing runs nearby. This coming winter I planned on doing my hardest marathon ever, on a course about three hours west of Sydney. The event is the Glow Worm Trail Marathon[2], and is in an area that could well be described as the middle of nowhere. Not really the middle of nowhere by Australian standards, but its certainly in a rather secluded area.

The closest town is Lithgow, an hour away, with a population of 21,000. Wallerawang (or Wang as it's known by the locals) is closer, but has less than 2000 residents. The entire Marathon is away from the townships in fairly rugged terrain. In fact, you need to drive about seven miles on a dirt road to get to the start line of the event. Cellular service is non-existent, and satellite phones work poorly due to the hills.

The area is so rugged that the Wollemi Pine that grows in the area was only know to exist in fossil records until 1994, and is now the only known survivor of a family of trees over 40m years old.

The event starts and finishes at Newnes, a place that these days consists of just a campground and a general store to support the campers. It has been like this following the dismantling of the railway line in 1940, and closure of the hotel in 1988 following a flood.

---

[1] A Marathon is officially 26.2 miles (or 42.195km) and a Half-Marathon – A race of 13.1 miles (or 21.1km). The distance of the marathon was standardized in 1921 based on the London Olympic Marathon of 1908. An Ultra-Marathon is any race longer than a Marathon.

[2] https://www.glowwormtrail.com/

The event itself consists of two distinct halves. The first half involves running up a hill, over it and down the other side towards Glen Davis, before returning to the start line the same way. The 13.2 miles[3] involves about 3894 feet of climb[4], none of which is on concrete or asphalt.

The second 13.2 miles is thankfully somewhat easier, with only 2345 feet of climb and decent. It also contains the reason for the event – an old rail tunnel about a third of a mile long inhabited by glowworms. That is over a mile of elevation up and a mile back down.

The thing is, I am a fairly good runner. Not the greatest, but I am normally in the top 20% in the races I enter. This year alone I have managed to get under 20 minutes for the 5km and under 3 ½ hours for the marathon. I worked out that running the 26 mile Glow Worm Trail Marathon would probably take me a bit over five hours, non-stop!

To give some perspective, this is the equivalent (if you could do it) of starting in the Yosemite Valley, running up El Capitan, and down the other side. And then turning around and doing the same thing again, all in the space of 26 miles! And if you think this sounds bad, I am already planning a 31 mile race next year, with 7800 feet of elevation[5]!

The entire event takes place in rain forest, in the middle of the Australian winter. I should note however that winter in this part of the world is a relative term, with the conditions likely to not go much below freezing, even overnight. Unfortunately, they likely won't get much above freezing for the entire race, but these are the conditions you prepare for.

Thankfully the conditions were not as bad as the run a friend did in Texas a couple of years ago whilst on holidays. This was a much shorter run, and she was wearing a half-gallon hydration pack filled with drinking water. During the run, the water froze solid. She tells me that event even surprised the organizers with how cold it was.

Given the harsh bush land and the remoteness, event communications are very important. Runners are known to push themselves too hard, not eat and to get injured during events. In a previous event, a runner was badly injured when a tree fell on her whilst she was running. Parts of the course are so steep that if they were any steeper and you would need to start rock climbing.

For the last couple of years, event communications has been handled by WICEN, the Australian Ham Radio emergency communications organization as a training exercise. This year they are aiming to improve the communications on the course by determining the best location for their repeater.

---

[3] Internationally most countries around the world predominantly use the metric system. Whilst Liberia and Myanmar are in the process of converting, the USA has not. As this article is targeted at readers predominantly from the USA, most measurements will be imperial. For reference there are about 5 miles in 8km, and 1000m is about 3281 feet.
[4] Climb – generally when runners talk about climb we only talk about the part going up. Since we normally start and finish in the same spot, we normally descend as far as we have climbed.
[5] That event is the Ultra-Trail Australia 50KM race in Australia's Blue Mountains. I did the 22KM race this year in 2h48m, but that only included 3900' of elevation climb

The great thing about this event is that there are a ton of spots where you can put a repeater. The bad thing is that testing the coverage is a chore. Even getting the repeater installed is difficult, and involves a 2-3 hour trek from the closest parking.

Coverage Site Survey

Over the Easter in mid-Autumn, WICEN wanted to try out a new spot for their repeater. This provided me with a chance to check out the course whilst providing reports of the radio coverage.

The idea was that we would run two parts of the course – we would warm up by running about half of the first half of the course, for a total of about 5-6 miles, and then run the second 13 mile half in full. Alas, things didn't turn out that way.

Unfortunately the guys from WICEN were unable to get to their preferred repeater position on top of the mountain. What looked good from the topographic maps and the satellite photos wasn't so good in real life. After some effort, they found they would need to climb up a 30 foot sheer cliff, which meant the radio test was no longer possible, and also ruled out that spot for a temporary repeater too.

The problem was that this area is not well surveyed, and the shadows on the satellite photos covered the sheer cliff. They mostly wanted to check the coverage on the second half of the course, meaning there was no need to run any further. You can check out where I did run on Strava[6]. I will definitely be heading out there at some stage to just explore the area.

Despite not achieving the aims of the test, I can at least talk about the equipment we were using out there, and how it works.

Radio Gear

As an article about Ham Radio, there is actually quite little to say about the radios we used. During the event WICEN are planning to use DMR digital voice repeaters, but during the test 2M FM was all we needed to use.

Both the WICEN team and myself were using 8W Baofeng dual band handhelds. These radios are inexpensive and did what they needed to do. As a bonus, in this remote area, they are unlikely to cause any interference on the off chance they are not as spectrally as pure as they should be. And Baofeng has been notorious, at least in the early days, for lacking rigorous engineering.

---

[6] https://www.strava.com/activities/2314605933

I had the opportunity to speak to one of the members of the radio team on a different event after visiting Glow Worm. They told me that for the Ultra-Trail Australia[7] (UTA) event they run multiple repeaters because of the terrain. In case of communications issues on course, they have a team of operators with 13 element UHF Yagi's they can point at the repeaters to provide event communications.

Garmin Forerunner 935

A few members of the running club are partial to the Garmin Forerunner 935 GPS watches. Part of this is that they were significantly discounted last Christmas, but they appear to be more reliable[8] uploading data than some other brands. The watch itself has a passive 64 color LCD display with backlight, GPS/GLONASS, along with heart rate, compass, altitude and cadence monitoring.

Also included are Bluetooth and WiFi. Bluetooth is used to connect to the mobile phone app, which sends data to the Internet should you need that. But Bluetooth has another function in this watch – when you are trying to get a GPS lock, the watch will attempt to use the phone GPS to determine the current position, and likely also to download the current GPS almanac showing the position of all the GPS satellites. This reduces the time to get GPS lock significantly, and dramatically improves battery life.

In this mode, I can log my runs with GPS data for well over 12 hours and probably closer to 24 hours per charge[9]. There is a mobile phone app that uses Bluetooth, although the watch also has WiFi built in. Bluetooth is more energy efficient, and is used to talk to the phone.

A comparison of watches whilst out running shows that a watch paired to a phone running in Low Power Mode takes a lot longer than a phone paired to a phone operating at full power. This means that whilst my watch will generally get GPS lock almost instantaneously, the other watch can take up to a minute to lock. This is likely because the watch downloads positional and GPS satellite almanac data from the phone reducing the time to fix.

One of the surprising advantages to this watch is that it uses physical buttons rather than a touch screen. Runners are often sweaty, and touch screens don't tend to work as well when the screen is wet. With physical buttons, it doesn't matter how wet things are. Whilst the manual suggests not

---

[7] a series of races ranging from 11-100km held in Katoomba, west of Sydney held in May each year. This is the largest sporting event in the Blue Mountains, with over 7000 people competing, and the third largest trail event in the world. https://www.ultratrailaustralia.com.au/

[8] reliable is an interesting word. Whilst mine has been rock solid, others managed to completely lock up a couple of times during running, resulting in loss of data. Thankfully after a restart, the watch is smart enough not to loose the data captured before it crashed, and will let the user continue as if nothing had happened.

[9] I will test this out at some stage. I am looking at a 60 mile race in a couple of years, and it will approach this time, although I do intend to charge the watch during the event with a portable battery.

using the buttons under water, us runners are crazy and will run in almost any conditions, day or night.

There are two features that are missing from this watch that are present on the higher end models. The first is that this unit does not permit the storage and playback of MP3 files. This can be an issue for some people – personally I just play MP3's from my phone, whilst my running companion has headphones that she has loaded MP3 files onto.

The other missing feature is the capability to use this watch as a Credit Card. Garmin has their own solution similar to ApplePay allowing people to pay by credit card using the Paywave non-contact RFID technology. In this case, the watch would use RFID to communicate with the credit card machine to authorize payments. Had it been present, I would need to enter the PIN number on my watch once a day, and whenever I took the watch off. Once again, I use my iPhone instead.

Mapping

One of the other cool features of this watch is the ability to upload a route to the watch, and it will warn you when you go off course. This is a really cool feature, and was rather useful on this run. When I went off course, I managed to work this out after a few hundred feet thankfully.

Alas, since this is a cheaper model, it doesn't have base maps showing streets and other features. As nice as they would be, when you are running off road, trails are often not well documented, even on web sites like OpenStreetMaps.

When I am following a route, I normally have it zoomed into a few hundred feet so I can work out where I am going relative to where I am. The watch does display a small arrow pointing at a direction, but this does not always work in areas of poor GPS coverage.

I did hear a report from the actual Glow Worm event where someone sabotaged one of the races, changing the markings on the trail causing some runners to take the wrong course. This is the first time I have heard of this happening during a race. Having the map on the watch whilst running at least warns you that something might be going on.

Having said that, I have also run in races where the organizers have changed the course at the last minute and forgotten to tell anyone, so no plan is foolproof. In another case, the marshals incorrectly located a turn around point in the Bangkok Half Marathon. They accidently added about 2½ miles to the course. All they could do after the event was re-issue the T-Shirts with the actual distance listed.

At this point I should point out what I mean by an area of poor GPS coverage. For GPS type devices to work properly, they ideally need to be able to see the sky right to the horizon in every direction. Alas, the human body tends to get in the way, but thankfully this isn't too much of a problem with modern devices. More of an issue is when there is terrain or buildings that restrict the 'view' of the sky.

When the view is restricted, GPS is not able to cope as well, particularly with reflected radio signals. This tends to increase the positional error. Thankfully in my experience with watches and phones has tended to be under a couple of hundred feet, and is normally a lot less.

The worst I have ever seen was about 15 years back with a GPS tracking unit I was developing and had mounted in my car. The car and GPS was under my carport, and the GPS 'lost it'. Whilst the car was safely parked, the GPS was reporting a journey of about 150 miles at a speed of about 300 miles per hour. What I think happened was that the software in the receiver failed and started tracking one of the GPS satellites by accident.

In many ways, the watch reminds me of 20 year old Garmin eTrex or Garmin 12 devices, and I suspect that there are still parts of that firmware living in this device.

If you do get lost, the watch has the ability to guide you back to the start, either by using the route you went or direct line of site. This can be a useful feature, although I haven't needed to use it yet.

Routes are loaded by copying a GPX file onto the device. There are some smart ways to do this wirelessly, but the watch also emulates a GPS thumb drive. I tend to use MapMyRun when designing routes and upload them wirelessly using http://dynamic.watch

If the run is one that someone has done before, as a premium Strava subscriber, I can download other peoples runs as a GPX file and upload it onto my watch.

I must say one of the best uses of the route functionality was last Christmas when I wrote the words 'Merry Xmas' in several hundred feet high letters running around the streets near where I live. This run was designed earlier in MapMyRun and the watch told me exactly where I needed to go.

Heart Rate Measurements

The watch measures the heart rate using physical properties, rather than the more common electrical ones, using a photoplethysmogram[10]. To do this it shines a green light onto the skin and monitors how much light gets absorbed. When the blood is flowing past, the amount of green light being absorbed increases. It can actually read the pulse not only in arteries, but also in subcutaneous tissues.

This does rely on the watch band being reasonably tight so that the back of the watch with the green LED is in constant contact with the skin.

Some runners prefer to use an electrical heart rate monitor around their chest that sends data to the watch via Bluetooth or the related ANT+ protocol. These can work well, although I have found in the past that my phone 'steals' the data from other people's heart rate monitors depriving them of the data. This appears to be related to the ANT+ protocol.

---

[10] https://en.wikipedia.org/wiki/Photoplethysmogram

Being able to monitor heartrate has some great health benefits. As a middle aged male runner, I am in the demographic that seems most prone to having a cardiac event whilst running. In fact, during the recent UTA22 race, my heart rate spiked about 20-25 bpm for a couple of minutes, before getting back to normal.

Based on this, I now have an alarm on the watch telling me whenever my heart rate is higher than it should be. Sometimes there might be a reason for it, but the alarm is a good indicator that I need to be careful.
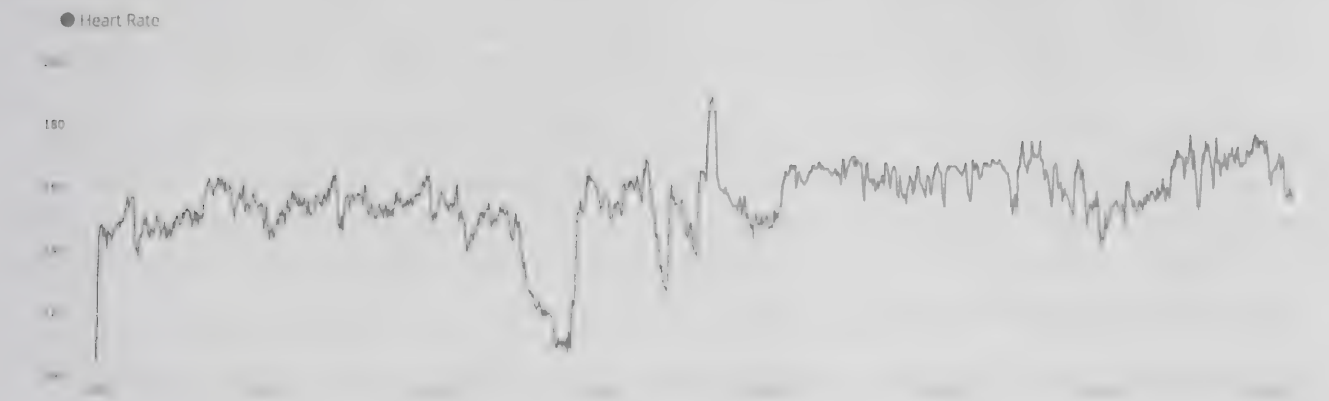
Since I wear the watch 24/7, it also keeps a track of my average resting heart rate. Depending on how much running I am doing, this is between 50 and 60 bpm. The more running I am doing, the lower the resting heart rate.

Sensor Fusion

Obviously devices such as watches have very small batteries, and need to save energy in any way they can. Anything with a radio transmitter or receiver in it will use up a heap of energy. So ideally, you want to have any radios turned off as much as possible. One of the major battery drains here is the GPS to determine the position and speed of the device.

GPS watch devices save energy by combining data from many sources. This is commonly known as Sensor Fusion[11]. This combines data from desperate sources with the aim of generating more accurate information in the process.

Data sources include three a dimensional compass, accelerometer and gyro, along with an air pressure sensor, giving a total of 10 degrees of freedom. 3D GPS data is also included giving a total of 13 degrees of freedom. Sensor Fusion would use this information to determine the following details:
- Cadence of the runner (Steps per minute)
- GPS Position
- Speed
- Elevation

---

[11] https://en.wikipedia.org/wiki/Sensor_fusion

One of the important things about sensor fusion is that each of the sensors needs to be measured at the exact same time wherever possible. This is particularly important for the compass, gyro and accelerometer. Essentially an integral is applied to the data, and if the points don't line up, there will be mismatches in the results.

Unfortunately, determining these details is not always that simple. Not all these data sources are available all the time. Pressure sensors often get blocked with water or dirt, particularly when running on trails. GPS receivers may only be turned on for a moment every few seconds.

Kalman Filters

And then there are the errors. Each of these sensors have errors, so the job of Sensor Fusion and Kalman Filters[12] is to determine what the most likely correct solution is, such that the data from each sensor is used, and that errors are minimized.

Think about the average runner. They will be running somewhere between five and 12 miles per hour, and will likely be moving horizontally. When moving vertically, such as on a ladder or stairs, their horizontal speed will be significantly slower.

With this in mind, algorithms and filters are then tuned to search for solutions mostly within a certain range from the last known good position. Using the accelerometer to determine the number of steps being taken is a great way to hone in on the correct result.

An easy way to think about this is by assuming that we know where we are now. If we can work out how many steps a minute the runner is taking, that gives us a fairly good idea where they will be in a minutes time. When you pull in the other data, even without an updated GPS position, you can make a fairly good guess as to the new position.

The guesses are not perfect. An example was a runner who discovered that when they work out on the treadmill that their distance was off by over 10%. This is probably because she takes smaller steps on the treadmill thanks to its incline. Thankfully the watch has a manual calibration mode, which tunes the internal filter. Once the unit was calibrated, it was accurate within about 1% of the correct distance.

This is an easy to understand example of tuning algorithms and filters. Things get a lot more complex when you escape the gym and get into the real world. Thus, one of the first real life applications of Kalman Filters was actually with the Apollo program to take man to the moon! And in case you are wondering, I don't understand most of the Kalman Filter Wikipedia page.

---

[12] https://en.wikipedia.org/wiki/Kalman_filter

Strava

Although I have mentioned it already in this article, I guess I should describe what Strava[13] is. In essence it is a social network predominantly for runners and bike riders, although other forms of physical activity are also included.

Rather than people posting photos of cute puppies (which I am sure must exist on the site), people upload the GPS traces of their runs and rides. These traces are timestamped so that people can see exactly how fast you went at every point, and where you went. It can also include details like heart rate and running cadence. Friends can then give you likes, or 'Kudos' for your runs.

But Strava goes further than that. Subject to privacy settings, it correlates your runs with others, and indicating who you ran with. Users can also route segments that are used as virtual races. After every run, Strava works out your time on the segments and reports on this. It also allows you to compare your time with others, creating a virtual race.

Like much technology, Strava does have significant privacy implications. They made the news due to their 'Heat Map' feature which allows you to zoom into any part of the world and see what the popular courses are. The problem was that Strava was being used by troops in the middle east. This was identifying the locations of those troops and where they trained.

Some Interesting Runs

Alas, Sensor Fusion does not always get things right. Below is the graphic from one of my runs earlier this year, where I was running up and down a hill[14]. The hill can clearly be seen, but each of the peaks and troughs should be the same as I was just going up and down the same hill. I also started and finished in the same location, so the start and finish of the plot should be the same height. The two are actually out by about 88 feet.

My guess is that this is related to a change in air pressure that happened late in the day. As a rough guide, air pressure decreases for every 30 feet of elevation by 1 hectopascal (hPa). Standard atmospheric pressure is 1013 hPa or 101.3 kPa.

Modern air pressure sensors have the resolution to determine climbing a single step, somewhere around 0.02 hPa. Whilst they have great resolution, their accuracy in determining elevation is not that great. The problem is not the sensors, but mother nature. These sensors can only read air pressure, and air pressure can change enough to cause issues. In this example, a low pressure change of 3 hPa over thirty minutes could have caused the issue.

---

[13] http://www.strava.com. You can see my Strava profile on the following URL
https://www.strava.com/athletes/11527827
[14] https://www.strava.com/activities/2265485986

These types of problems are generally corrected by calibrating the pressure sensor using GPS altitude as well as other information sources. As can be seen here, this is not always successful.

	Pace	GAP	Heart Rate	Cadence	Temperature
Avg	11:20 /km	7:23/km	129 bpm	142 spm	24 °C

Another example shows a comparison of my pace doing hill repeats compared to the hill[15].

In this case, the hill does look like an almost perfect triangle wave. The pace looks like a rather noisy square wave, which is what you expect. I got slower every time I ran up the hill, but I generally ran faster going down the hill than running up.

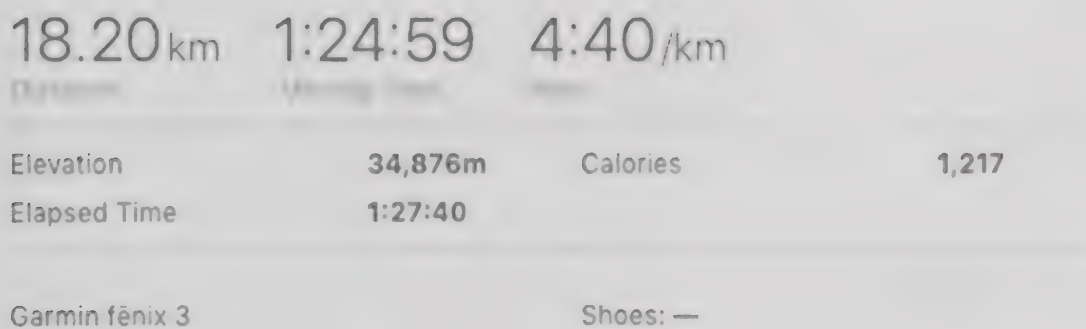|	Pace	GAP	Heart Rate	Cadence	Temperature
Avg	7:19 /km	6:30/km	137 bpm	171 spm	28 °C

---

[15] https://www.strava.com/activities/2214282641

More interesting is the comparison of heart rate to elevation. In this case, the heart rate is an almost perfect square wave. These are the sort of effects you would expect doing signal analysis with integrals and derivatives.

	Pace		Heart Rate		Temperature
Avg	7:19 /km	6:30/km	137 bpm	171 spm	28 °C
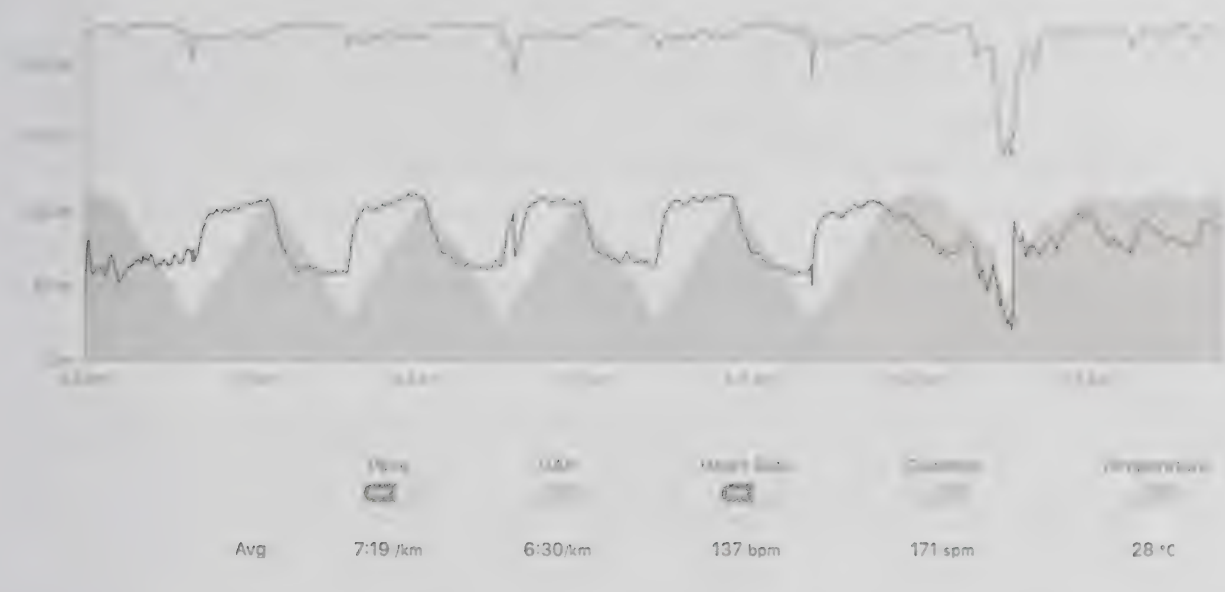
In case you are wondering, these runs were on an old dirt road near where I live called Old Ford Road. It is about 130 years old, and was built as an unemployment project, making it older than the nation on which it stands. As such, it is likely one of the oldest intact roads in the country. These days access is restricted to foot and bicycle traffic, and the road no longer goes any further than the bottom of the hill due to a military base.

I have found two runs on Strava from other runners that give an example of what happens when things go wrong. In this first example the athlete ran just over 11 miles. The watch believes he climbed 114,000 feet during this run.

18.20 km   1:24:59   4:40 /km

Elevation	34,876m	Calories	1,217
Elapsed Time	1:27:40		

Garmin fēnix 3   Shoes: —

In this case, the watch was an older Fenix 3, which was released two years before the ForeRunner 935, in early 2015. Sensor technology has improved significantly in those two years, as has the processing power available within a device like a watch.

In this case the likely cause of the obviously incorrect elevation is a blocked air pressure sensor on the back of the watch. Cleaning the sensor will likely fix results like this one. It is interesting to speculate what happened in the graph below about the 8km (5 mile) mark.

Before then, the results were high, but there was some variation. Afterwards the graph appears to go to zero and then when it returns, does not change for the entire race. It is possible that either the sensor got blocked further at that time, or possibly the watch knew that there was an issue, and tried to make sense of the results. Whatever the cause, the results are obviously wrong.
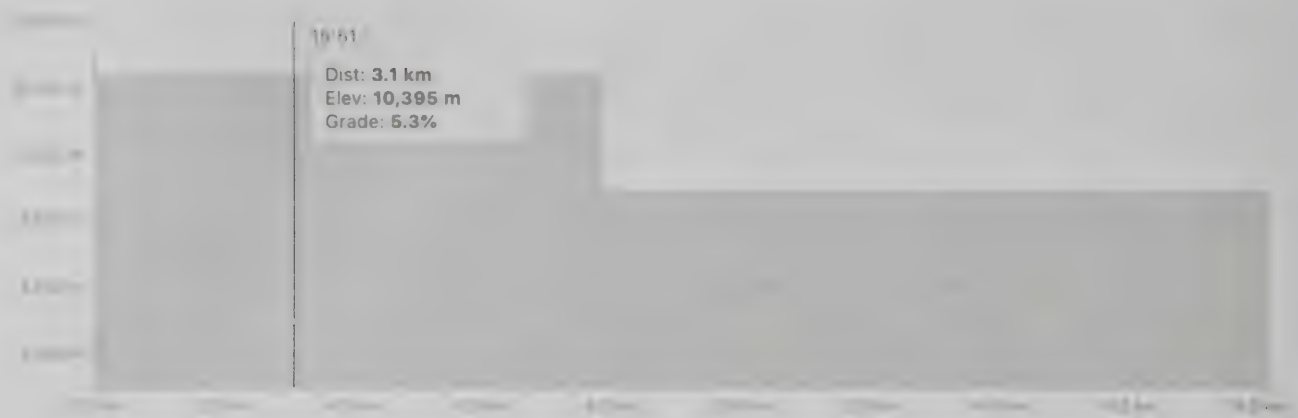
Another example appears below. What is interesting in this case, apart from some 'brick wall' changes is the exponential curve with a long time constant. This could relate to a partial blockage, or could suggest an algorithm attempting to fix erroneous results.

Mountain Bike Riding Too

You might be wondering what happens if someone is riding a bike rather than running. Well, there are techniques to get the best results. When you record an activity, the first thing the watch does is ask what sort of activity you are recording, be it a ride or a run, or even a swim.

If you select a ride, the watch will not be trying as hard to determine the number of footsteps that you are making, and will also assume that you will be able to go a bi rather on wheels than running. This is not always a case, but is a reasonable assumption.

The good thing about riding is that you are generally covering more distance, so the errors in reading from GPS tend to be comparatively lower.

Once again, you can get accessories for the watch to improve the accuracy of the readings. One common sensor is worn on the top of the shoe with a 9 or 10 degree of freedom sensor that helps to determine the number of revolutions per minute you are peddling. Another sensor is similar but goes on the pedal, and also measures how hard you are peddling. This allows the software to calculate the amount of energy you are burning.

Whilst we are on the subject of bikes, video games are definitely invading training. Adapters are available for bikes that let you ride in front of a television on bike, competing with other people also in their own homes. What makes this even more interesting is that the adapter forces you to pedal harder going uphill and allows you to coast going downhill.

This makes 'riding' in exotic locations like Paris, London or Washington DC easy. It did however surprise a friend the first time when his colleague checked Strava before work and saw his workmate riding in London before an important meeting.

APRS Whilst Running

One thing I really haven't got into trying is using APRS whilst out running. It is something I do want to try, but I haven't got around to it yet. Part of this is that many of the areas I run don't have very good APRS coverage. I can sometimes fix that by using my car's APRS rig as a digipeater, but in most cases, the increase in coverage would be minor.

I have done a few runs running APRS on my iPhone, but this tended to be on road races, rather than areas with poor mobile coverage. In areas of poor mobile coverage, phone batteries tend to die more quickly, and I would prefer to save the battery in case I need to make an emergency phone call.

I do have plans to carry an APRS rig with me next year when I do the UTA 50km race. This will require some planning, and I will need to consider if I need to carry spare batteries. The weight of the APRS gear is not really a problem, as most trail races already require me to carry up to two liters of water along with other mandatory gear. In this case, the weight of the tracker is minor compared to the weight of everything else.

Other Technologies

Of course, these devices log using GPS, but need some type of connectivity to upload their data for wider distribution.

In the sports tracking world, there are two options available for disseminating GPS data to the wider world, similar to how APRS works. The first is the SPOT Tracker, that uses the Globalstar satellite network. The second are the Garmin inReach devices that use the Iridium satellite network.

In both cases, the device will transmit position reports every 2-10 minutes via satellite before being disseminated via the Internet. Both devices require subscriptions costing about 50% of the hardware cost per year. These units tend to use satellites in LEO, or Low Earth Orbits, as a way to reduce the costs launching the satellites.

The Actual Glow Worm Trail Marathon

I was hoping to get this article finished in the days after the site survey. Unfortunately, as you might be able to work out, this article just kept getting longer and longer and longer. There was more information that I wanted to put it. Then, when I was almost ready, I realized that it was only two weeks until the actual event, so I might as well just wait for it before getting it published.

One of the things with running is that it does take a toll on the body. Doing these type of events is even worse. One rule of thumb is that the recovery time from races is about one day per mile. Thus, a 26 mile event requires almost a month of recovery time. This is not to say that you can't run in that time. More that the body will not be back to peak performance until then.

For me, I am finding that my body recovers enough to be able to actually run fairly quickly, particularly after road races. What I have found is that after marathons, my endurance takes longer to recover. Whereas, I might normally be able to do a 10-15 mile training run at will, I find that I get exhausted after running half that.

Doing events with lots of up and down climbing takes the toll on the legs too. This generally comes in the form of DOMS, or Delayed Onset Muscle Soreness. You might feel fine the day after a big race, but then it will hit you, and going up a single set of stairs will be torture. I am looking at a competing in a race in two years' time with about 15,000 feet of elevation. I don't expect to be able to walk for a week!

The reason I am mentioning this is that after a club 10 Mile[16] handicap[17] race two weeks before the Glow Worm Marathon I managed to injure my quads and adductor in my leg. Treatment helped, but

---

[16] This race is actually 10 miles long. The race dates back to before Australia went metric. Thankfully, 10 miles translates almost exactly to 16km, making it easy to measure in a metric country.

[17] In most races, the fastest person wins the race. In a handicap race, you are given a time which the handicappers feel meets your ability. In this type of race, the person who beats their handicap time by the most wins the race. In this way, a fit 20-year-old can be beaten in a race by a retiree with arthritis.

in the end, I decided that running this marathon had the potential to cause my injury to get a lot worse. With great reluctance, I decided to pull out.

So, rather than do a tough marathon, I decided to do an easier local trail run[18] instead. Thankfully, this run went well, and I can now resume my training for the multitude of events I have planned for the coming months. I ran just over 15 miles with about 1200 feet of climb in a bit under three hours in wet conditions. Given that my fitness had deteriorated in the last few weeks, this was a good result.

Conclusion

I would like to thank those who helped with this article, either by running with me, assisting with the site survey or proof reading. They know who they are, and I could not have written this article without them.

As for me, I have a few road and trail events lined up. Next year I hope to run the 28 mile Six Foot Track Marathon just west of Sydney if I get in. The Marine Corp Marathon in Washington DC looks interesting, but we will see when I do that. I am also looking at doing the Ultra-Trail Australia 50km race in May 2020, which will definitely be a challenge. Thankfully I have time to train for that.

---

[18] https://www.strava.com/activities/2453503488

Notes